MODELING, SIMULATION, AND ANALYSIS OF A DECOY STATE ENABLED QUANTUM KEY DISTRIBUTION SYSTEM

THESIS
MARCH 2015

Ryan D. L. Engle, Captain, USAF
AFIT-ENV-MS-15-M-181

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-15-M-181

Disclaimer

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT-ENV-MS-15-M-181

MODELING, SIMULATION, AND ANALYSIS OF A DECOY STATE ENABLED QUANTUM KEY DISTRIBUTION SYSTEM

THESIS

Presented to the Faculty

Department of Systems and Engineering Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirement for the

Degree of Master of Science in Systems Engineering

Ryan D. L. Engle, BS

Captain, USAF

March 2015

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENV-MS-15-M-181

MODELING, SIMULATION, AND ANALYSIS OF A DECOY STATE ENABLED QUANTUM KEY DISTRIBUTION SYSTEM

Ryan D. L. Engle, BS
Captain, USAF

Committee Membership:

Michael R. Grimaila, PhD, CISM, CISSP
Chair

Douglas D. Hodson, PhD
Member

Gerald Baumgartner, PhD
Member

Abstract

Quantum Key Distribution (QKD) is an emerging technology which uses the principles of quantum mechanics to provide unconditionally secure key distribution. QKD systems are unique in their ability to detect the presence of an eavesdropper and are being marketed for use in applications where high levels of secrecy are required such as in banking, government, and military environments. However, QKD technology has not gained widespread acceptance, primarily due to the newness of the technology. Research has shown that practical QKD system implementations differ significantly from their theoretical designs raising questions about their suitability for use in sensitive cryptographic applications. There is a critical need to model, simulate, and analyze QKD systems in order to understand the impact non-ideal components and practical implementations have on performance and security. QKD systems are complex systems composed of electrical, optical, and electro-optical components. Their design and analysis requires expertise across multiple disciplines including computer science, computer engineering, electrical engineering, information theory, optical physics, and quantum physics. The inherently multi-disciplinary nature of QKD systems makes it an ideal candidate for study using Model Based Systems Engineering (MBSE) Processes, Methods, and Tools (PMTs).

This thesis presents a study of the QKD decoy state protocol using MBSE PMTs. Specifically, the modeling, simulation, analyses, verification and validation of a decoy state enabled, BB84, prepare and measure, polarization based QKD system is evaluated in the presence of an eavesdropper implementing a theoretical photon number splitting attack. The primary goal of the research is to gain improved understanding of the operation and performance of the QKD decoy state protocol. This objective was achieved by examining the relationships between design, implementation, and operational choices with respect to the decoy state protocol. The main contributions of this research include the development of a decoy state protocol model, validation of the decoy state protocol in a QKD system model implementation, and confirmation that application of MBSE PMTs are critical to the understanding and analysis of complex systems. This thesis presents the first known application of MBSE PMTs to define and analyze a decoy state enabled QKD system and provides utility to system developers, designers and analysts who seek to quantify QKD system performance and security.

Dedication

This work is dedicated to my grandparents and my mother who instilled integrity in me and always inspired me to do my best. Additionally, I would like to thank my wife and children for their unwavering support and encouragement—without you, I would be lost and unbalanced.

Acknowledgments

This thesis project would not have been possible without the support and guidance of the QKD research team. First and foremost, I want to thank my advisor, Dr. Michael Grimaila, for allowing me to join this thrilling project, consistently pushing me to perform at a high level, providing mentorship and encouragement, and fighting for me to continue supporting this research as an in-residence PhD student. Dr. G, without your support my career would have a very different and less rewarding path—thank you. Next, I would like to thank Major Logan Mailloux who provided daily encouragement, military mentorship and comradery, and taught me that "every problem is a solution."

I would also like to thank Dr. Douglas Hodson who taught me the power of integrating statistics, experimental design, and software engineering which enabled me to "finish" this research. I also owe a thank you to Dr. Gerald Baumgartner for his guidance and the opportunity to participate in this exciting area of research. Additionally, I would like to thank Rob Cernera for his assistance while learning about the capabilities of OMNeT++.

Finally, I would like to thank Dr. Timothy Lacey, John Phillips, and Ryan Harris for their support in using the Cyber Defense Exercise servers to execute my simulations.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Terms

AFIT	Air Force Institute of Technology.
APD	Avalanche Photo Diode.
Architecture	“High-level design and execution structure of a software system. A software's architecture is analogous to a building's architectural choices for a foundation, frames, window locations, etc” [1].
BB84	Bennett and Brassard’s quantum key distribution protocol defined in 1984 [2].
BSE	Beam Splitter Experiment.
Block size	A positive integer representing the number of bits in the block.
CD	Classical Detector (optical component module in qkdX).
Classical channel	A channel over which classical network communications occur.
Coherent pulse	An optical pulse with a fixed phase relationship between the electric field values at different locations or at different times.
COTS	Commercial-Off-The-Shelf.
CPG	Classical Pulse Generator (module in qkdX).
Detection limit	The number of pulse detections that occur at Bob before Bob tells Alice to stop sending pulses.
DoD	Department of Defense.
DoDAF	Department of Defense Architecture Framework.
DSE	Decoy State Enabled (as in DSE QKD).
DSG	Decoy State Generator (module in qkdX).
DT&E	Developmental Test & Evaluation.

Error threshold The maximum error rate acceptable by the BB84 protocol. An error rate above this value will cause the sequence to terminate due to suspicions that an eavesdropper is intercepting qubits.

Experimental Unit Item in an experiment being studied [3, 4].

Frame A collection of optical pulses generated by a sender containing a bright pulse and one or more weak pulses with defined temporal spacing designed to provide the ability for a receiver to synchronize on the pulses.

Framework "A set of source code or libraries which provide functionality common to a whole class of applications. While one library will usually provide one specific piece of functionality, frameworks will offer a broader range" of functionality [5].

Gain A measure characterizing a channel between a sender and a receiver calculated by dividing the number of detections at the receiver by the number of pulses emitted by a sender (e.g.,$\frac{\#\,Detections}{\#\,Pulses\ Sent}$) [6].

GUIDEx Guide for Understanding and Implementing Defense Experimentation [7].

IDE Integrated Development Environment.

IDEF Integrated DEFinition.

INCOSE International Council on Systems Engineering.

LTS Laboratory for Telecommunication Sciences.

M&S Modeling & Simulation.

MBSE Model Based Systems Engineering.

MPN Mean Photon Number. The statistical average number of photons contained within a pulse over a specified interval of time. [8] Typically modeled using a Poisson distribution.

NED file NEtwork Description. OMNeT++ file type to define, connect, and assemble modules [9].

NRL	Naval Research Laboratory.
OMNeT++	Open source simulation framework.
OPM	Output Power Module (module in qkdX).
OSL	Optical Security Layer (module in qkdX).
OT&E	Operational Test & Evaluation.
OV	DoDAF Operational Viewpoint.
Partners	Secret key users and their subsystems.
PD	Polarization Detector (module in qkdX).
Pulse	A burst of optical electromagnetic energy with finite duration.
PM	Pulse Modulator (module in qkdX).
PMT	Processes, Methods, and Tools.
PNS	Photon Number Splitting (Attack).
QBER	Quantum Bit Error Rate.
QKD	Quantum Key Distribution.
qkdX	QKD eXperimentation modeling framework.
QND	Quantum Non-Demolition.
Qubit	Encoded single photon that is used to represent a particular bit value [10]; concatenated version of *quantum bit.*
Response	Output of the experiment [11].
Run	See *trial*.
RUP	Rational Unified Process.
SADT	Structured Analysis and Design.
Sampling unit	See *experimental unit*.

Scenario Sequence of events through a use case [12].

SE Systems Engineering.

Sifted Detection A detection at Bob where Alice and Bob used the same basis [6].

Slot A defined time interval contained within a frame of optical pulses.

SME Subject Matter Expert.

SPD Single Photon Detector.

SV DoDAF Systems Viewpoint.

SysML Systems Modeling Language.

TA Timing Analyzer (module in qkdX).

Toolbox A set of related software components in which components may be modified, used together, or connected to each other to produce other components, functions, or behaviors.

Toolkit See *toolbox*.

TPG Timing Pulse Generator (module in qkdX).

Treatment Intentional change made to input variables by an experimenter on an experimental unit [13, 3].

Trial An "observation of the experimental unit under treatment" [11, 13].

UML Unified Modeling Language.

V&V Verification and Validation.

Yield (Y_n) The conditional probability of a detection event at Bob's side given that Alice sends out a pulse [14].

WCP Weak Coherent Pulse. A highly attenuated coherent pulse with a Mean Photon Number approaching sub-photon levels. Also called a WP.

WDM Wave Division Multiplexor (optical component module in qkdX).

AFIT-ENV-MS-15-M-181

MODELING, SIMULATION, AND ANALYSIS OF A DECOY STATE ENABLED QUANTUM KEY DISTRIBUTION SYSTEM

I. **Introduction**

Motivation and Background

QKD is an innovative cryptographic technology designed to provide *unconditionally secure*[1] key distribution between two parties. QKD technology exploits the fundamental properties of quantum mechanics to generate and distribute shared secret keys and can be used in applications where high levels of secrecy are required such as those found in banking, government, and military environments. QKD is unique in its ability to detect the presence of an eavesdropper attempting to subvert the secure distribution of key material. However, practical implementations of QKD systems differ significantly from their theoretical designs due to the non-ideal components used to construct real-world systems. Due to the complex nature of the system, the non-idealities, and multitude of implementation variants, there is a critical need to model, simulate, analyze, and understand the impact non-ideal components and system implementations have on performance and security of realized QKD systems [10, 15, 16]. To help address this critical need, researchers at the Air Force Institute of Technology have developed a QKD eXperimentation (qkdX) modeling framework [17]. The research presented in this thesis seeks to enhance the qkdX framework by building and validating new capabilities within the framework.

Perhaps the most notable non-ideality in practical QKD systems is the use of classical laser sources to generate optical pulses for key generation [18]. Because on-demand single photon sources are not available using current technology, classical laser pulses are generated and attenuated down to “weak

[1] In this context, “unconditionally secure” means that an adversary, with unlimited computing power, can only obtain a negligible amount of information about the distributed secret key without introducing detectable errors [2, 10, 15].

coherent pulses" with sub-photon Mean Photon Numbers (MPN) with photon distributions that follow a Poissonian distribution. This means, for example, in an attenuated laser pulse with an MPN of 0.1, approximately 90% of the pulses will have zero photons, 9% will have one photon, and 1% will have more than one photon. Each multi-photon optical pulse, introduces critical vulnerabilities into the unconditionally secure QKD process. Specifically, these multi-photon pulses enable an eavesdropper to gain information on the secret key without detection through the Photon Number Splitting (PNS) attack [19, 20, 21, 22, 23]. In 2003, the decoy state protocol was proposed to mitigate the PNS vulnerability; however, its implementation has not been studied in detail nor characterized for realized systems [24, 15]. This example demonstrates that ability to rapidly and accurately model real-world, non-ideal laser pulses and analyze the effectiveness of the decoy state protocol is of paramount importance to fully understand QKD system performance and security.

Real-world QKD systems contain electrical, optical, and hybrid components, and their design and analysis requires expertise across multiple disciplines including computer science, computer engineering, cryptography, electrical engineering, information theory, optical physics, quantum physics and statistics. The inherent, complex, multi-disciplinary nature of QKD technology makes it an ideal candidate for study using Model-Based Systems Engineering (MBSE) methodologies [25, 26].

Problem Statement

While the decoy state protocol was developed to mitigate the photon number splitting attack, there has been no published research on the modeling and simulation of the vulnerability to evaluate the effectiveness of the protocol. There is a critical need to understand the impact that decoy state protocol has on the performance and security of QKD systems.

Research Goals and Objectives

The purpose of this research is to study the decoy state protocol commonly used in QKD systems through the use of an MBSE methodology consisting of Processes, Methods, and Tools (PMTs).

Specifically, this research focuses on the modeling, simulation, and analysis of a decoy state enabled, BB84, polarization encoding, prepare and measure QKD system in the presence of a theoretical eavesdropper implementing a photon number splitting attack. The overall goal of research is to develop an improved understanding of the operation and performance of the QKD decoy state protocol and provide a working simulation model. The primary objective of this research will be achieved through the design, implementation, verification, validation, and performance analysis of a decoy state enabled QKD system model using an MBSE approach. The results of the study will enable an analyst to gain a deeper understanding of the relationships between performance and design choices within the decoy state enabled QKD system.

Research Questions

The research objectives will be accomplished by answering the following high level research questions and their sub-ordinate research questions:

RQ1. What is Model-Based Systems Engineering and why is it important?

RQ1.1 What MBSE Processes, Methods, and Tools (PMTs) are useful to a modeler in defining, implementing, verifying and validating a QKD system model?

RQ2. What is Quantum Key Distribution (QKD)?

RQ2.1 What is the BB84 protocol?

RQ2.2 What is a Photon Number Splitting (PNS) attack?

RQ2.3 What is the Decoy State protocol?

RQ3. How effective is the Decoy State Protocol at detecting the Photon Number Splitting (PNS) attack?

RQ3.1 What measurements are necessary to detect an eavesdropper performing a PNS attack?

RQ3.2 What behaviors should be captured in a decoy state enabled QKD system model?

RQ3.3 How flexible is the qkdX framework for building and studying fit-for-purpose models?

RQ3.4 What changes need to be made to the existing qkdX simulation framework for studying a decoy state enabled QKD system model?

Methodology

Overall, this research is guided by a MBSE approach to model building and experimental design. Pertinent processes, metrics, and operational parameters related to the QKD system operation and the decoy state protocol is extracted during a literature review. Background material is condensed to define fully dressed use cases, activity diagrams, and timing sequences. These products are captured using the Department of Defense (DoD) Architecture Framework (DoDAF) viewpoints to further define, understand, and capture the required capabilities [27]. Use cases are used to document and define a set of model behaviors supported by architecture views and traceable requirements. These formalized requirements are used for model Verification and Validation (V&V) through the comparison of published experimental results and empirical data. A decoy state protocol model is implemented using the qkdX modeling framework and its behavior will be verified. Lastly, the model is used to conduct performance characterization to more fully understand the decoy state enabled system's performance and security.

Assumptions and Limitations

As in any research endeavor, the research findings presented in this thesis are subject to several assumptions and limitations including the following:

1. BB84 is the name given to the first QKD protocol outlined by Charles Bennett and Gilles Brassard in 1984 [2]; however, there is no specific implementation standard. While this protocol was defined for over 30 years at the time of this study, there have been no formal system engineering decompositions of the BB84 protocol explicitly documenting its behaviors. Thus, system implementations

widely vary as many design decisions are left to QKD system developers. As a result, the author was required to make and document critical design decisions in order to build the use cases used in this research.

2. The decoy state protocol presented in this research does not have a fixed reference implementation. This research makes use of the decoy state protocol as first proposed and demonstrated by Lo *et al.* [28] which uses a signal state, a decoy state, and a vacuum state [14]. While other researchers have proposed additional states, implementing two decoy states requires the least complexity and provides the same capability to detect an eavesdropper as an infinite number of decoy states. Additionally, the decoy state security condition enables PNS detection by comparing either yields or errors; this study focused on the yield comparisons as a first step in modeling research [28, 29]. Other variations of the decoy state protocol were not studied in this research effort.

3. Two important assumptions have been made related to the communications medium that connects the two authorized parties, often called Alice and Bob in the literature. First, they must know the properties of their communications medium very well [28]. For example, Alice and Bob must jointly know their normal channel efficiency because it is a critical factor in the secure operation of a QKD system. The second assumption is that this knowledge of the communication medium excludes the presence of an active eavesdropper. These assumptions enable Alice and Bob to establish a set of tolerances (i.e., bounds for the signal and decoy yield estimates) which are required to operate a practical system in a secure manner [28]. Both of these ideas will be examined further during the discussion of the decoy state implementation.

Implications

This research demonstrates the ability to design and implement a decoy state enabled QKD system model within the qkdX modeling framework. The results and lessons learned will enable system developers, system designers, analysts and future model builders to more efficiently use the qkdX framework and suggest opportunities for improvements or extensions.

More generally, the promise of unconditionally secure communications is of strategic interest to the United States and other nations around the world. When QKD is combined with the One-Time-Pad algorithm to encrypt messages, it results in unbreakable communications [30, 31]. Therefore, it is of vital interest to study the performance and security impact of non-idealities in realized QKD systems. The first nation to develop reliable unbreakable communications will have a substantial advantage over its adversaries as history has demonstrated [32]. Research that supports improvements or shapes practical implementations should be of utmost interest these parties. The United States Air Force and Department of Defense investment in advanced academic degrees and research related to secure communications has the potential to result in significant advances to national security.

Thesis Outline

The remainder of this document is organized into seven chapters followed by appendices. Chapter II outlines background information necessary for the reader to understand the MBSE approach used to model and study decoy state enabled QKD systems. Chapter III explains the research methodology and discusses how the research questions presented in Chapter I are answered. Chapter IV presents a journal article, submitted to the *Journal of Defense Modeling & Simulation*, which presents our initial decoy state enabled QKD model. Chapter V presents a journal article, submitted to *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, which outlines the approach and results of applying the MBSE methodology to define, design, implement, verify, and validate the full decoy state enabled QKD model. In Chapter VI, the results of the modeling process and experimentation are analyzed. In Chapter VII, conclusions are drawn, research questions are answered, lessons are captured, and future research is proposed. Additionally, details of the research and experimentation are provided as a series of appendices A-F.

II. Literature Review

Overview

This chapter is divided into three sections which cover principles of Systems Engineering (SE), Model-Based Systems Engineering (MBSE), and Quantum Key Distribution (QKD). Specifically, the SE section covers the relevant theory upon which MBSE is built. Likewise, the MBSE section explains the relationship between SE and MBSE and outlines the elements of an MBSE methodology. Finally the last section provides an overview of QKD theory, limitations, and the qkdX modeling and simulation framework. The information presented in this chapter provides a foundation for understanding and conducting the research presented in this thesis.

Systems Engineering

Key Definitions

The International Council on Systems Engineering agrees that establishing shared terminology facilitates effective communication and is a primary task for systems engineers [33]. In spite of this assertion, neither "system" nor "system engineering" has a universally agreed upon definition [25]. This creates some ambiguity in understanding but facilitates flexibility or tailorability of application. The author's chosen definitions are based on what he believes to provide the best clarity from his course of study and discussion of his research.

A system can be defined as follows:

1. "[A] group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose" [16, p. 12],
2. "[A]n assemblage or combination of functionally related elements or parts forming a unitary whole" [25, p. 3],

3. "The organization of hardware, software, material, facilities, personnel, data, and services needed to perform a designated function with specified results, such as the gathering of specified data, its processing, and delivery to users" or
4. "A combination of two or more interrelated pieces of equipment (or sets) arranged in a functional package to perform an operational function or to satisfy a requirement" [34], and
5. "[A]n integrated set of elements, subsystems, or assemblies that accomplish a defined objective" [33, p. 5].

As the reader can ascertain, these definitions are largely similar and they share the idea that systems are made of inter-related parts that when combined serve a purpose or function. This generalized definition of system will be used consistently throughout this study.

Similarly, Systems Engineering has a number of related definitions:

1. "[A]n interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem" [33, p. 6, 35] and
2. "[A] methodical and disciplined approach for the specification, design, development, realization, technical management, operations, and retirement of a system" [36].

Just as the multiple definitions of "system" were not mutually exclusive, neither are those for "systems engineering." Blanchard and Fabrycky, themselves, acknowledge there is no universally agreed upon definition for the subject. However, they suggest that some commonalities exist which include: 1) an approach that views the system as a whole, 2) consideration of the system's entire life cycle, 3) characterization of system requirements and 4) the notion that an interdisciplinary approach is necessary [25]. While each organization may focus on a unique nuance within the SE discipline, their definitions tend to holistically agree with each other.

An important definition related to both systems and SE is that of system of systems. A system of systems is a system composed of elements which independent meet the definition of a system [37, 33, 38]. While this definition may not appear profound, it does establish a clear delineation between a complex system and a system of systems.

Concepts

This section attempts to define a number of ideas that appear disjoint at first glance; however understanding is necessary to synthesize ideas described by SE development processes and MBSE principles. The system life cycle is described first and identifies phases through which a system will pass its existence. Next, the SE concept of requirements is defined and the importance is discussed. Verification and Validation (V&V) and their relationship to requirements are discussed afterward. Lastly, system decomposition and integration are defined and compared.

Life cycle

A system's life cycle begins from the identification of a need or definition of a problem and lasts through the system's retirement or disposal [25, 36, 39, 33]. While each organization surveyed agrees with the definition of life cycle, the identification of phases (or stages) within a system's life cycle varies for each organization. For example, Blanchard and Fabrycky identify the following five phases: 1) conceptual design, 2) preliminary design, 3) detailed design and development, 4) production/construction, and 5) operational use and system support [25]. Alternatively, INCOSE identifies the following seven stages: 1) exploratory research, 2) concept, 3) development, 4) production, 5) utilization, 6) support and 7) retirement [33]. Still the Department of Defense identifies five different phases: 1) materiel solution analysis, 2) technology development, 3) engineering and manufacturing development, 4) production and deployment, 5) operations and support [40]. Each methodology defines a set of activities which shape the growth of the system of interest from immature concept to ultimately the non-existence of the system.

Figure 1 provides a generic model of the life cycle phases from concept to retirement [33]. This model illustrates how the stages of a life cycle model may progress over time. The phases exist to delineate work efforts or to serve as a means of delineation for tailored process management [25].

Understanding in which phase a system project exists will shape the purpose of any study, e.g., model development, as well as influence which process model is followed [39]. Process models are explored in the next section.

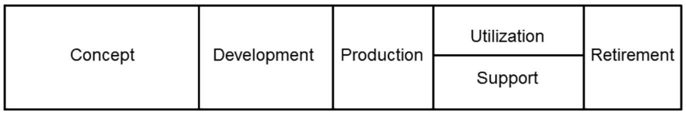


Figure 1. Generic Life Cycle Model [33].

Requirements

Requirements are an important component of SE; they provide a vehicle to define the significant desired behaviors of a system model while facilitating traceability during integration and Verification and Validation (V&V) to meet user needs [25]. Requirements also shape the purpose of the model and affect the level of abstraction and complexity of the implemented design. Banks *et al.* argue for beginning with a simple model and adding complexity as well as stating that "only the essence of the real system is needed" instead of developing a model that maps perfectly to a real world system [16].

Three of the 16 processes in the DoD SE Process Model, i.e., Stakeholder Requirements Definition, Requirements Analysis, and Requirements Management, focus directly with requirements. Five others, Configuration Management, Technical Data Management, Interface Management, Verification, and Validation also involve system requirements [41, 36, 37]. Additionally, Blanchard and Fabrycky discuss the importance of requirements throughout all phases of the SE process [25]. Furthermore, Gomaa discusses quality attributes of requirements such as correctness, completeness, consistency, verifiability, and traceability, which are applicable to all 16 SE processes [12].

Verification and Validation

Verification and Validation (V&V) are two phases in the systems engineering process that ensure the system is built correctly and meets the identified user needs. V&V is often associated with system test and evaluation, and is considered part of the DoD realization phase of systems engineering [25, 41]. In fact, the DoD identifies each as phase as having its own separate technical process. The DoD also delineates verification as a Developmental Test & Evaluation (DT&E) activity and validation as an Operational Test & Evaluation activity (OT&E) [36, 41].

In system development, verification is a process that confirms that realized system conforms to system requirements [36]. Verification answers the question “was the item built right?” In software-based applications, verification also includes debugging. In simulation modeling, verification confirms that model was built correctly [16, 42, 43, 44].

At a systems level, there are four distinct types of verification: inspection (or examination), demonstration, testing (or evaluation), and analysis. Inspection generally consists of tactile measurement and is usually nondestructive. A visual observation or physical measurement that the item conforms to a specification are examples of inspection or examination. Demonstration involves the operation, adjustment, or configuration of the system or subsystems. Testing is an activity that assesses the system’s conformance through controlled conditions and normally involves the utilization of scientific principles and procedures. Analysis may be conducted through the use of modeling and simulation or other mathematical or analytical methods to measure the system’s compliance with requirements [45, 46].

Validation, on the other hand, is a process, which in systems development, confirms the correct system was built to meet user needs [36]. Validation answers the question “was the correct item built?” In simulation modeling, the goal of validation is to assess whether the real system is correctly, to the necessary resolution, represented by the model [16, 42, 43, 44].

Decomposition and Integration

Decomposition is a process by which functions or system behaviors are broken down into smaller units of mutually exclusive activities from the top down. Decomposition may be described in terms of functions, activities, logic, structures, or etc. As such, decomposition is not unique to SE, but it is a concept that must be understood prior to a discussion of SE development process models [47, 33]. In contrast to decomposition, integration is a process by which functions or system behaviors are unified into larger system elements from the bottom up [33]. Obviously, decomposition and integration are two sides of the same coin. Both will be discussed further in the next section.

Development Process Models

A number of development process models exist to frame the work effort in each life cycle phase required to realize, operate, and support systems. The waterfall, spiral development, incremental development, the Rational Unified Process, and the "Vee" process models are commonly employed in SE [25, 12, 33]. Each model will be briefly examined in the following sections.

Waterfall Model

The waterfall model, introduced in 1970 by Dr. Winston W. Royce, contained seven steps or phases to facilitate management of software development. The waterfall method involved non-overlapping activities that were to be completed prior to advancing from one phase to the next. This model is suitable for projects with requirements that are stable, i.e., non-changing, during the development process. Contrary to popular belief, Royce discussed returning to previous phases as necessary. One limitation of the waterfall approach is that no system or capability is available to users until the end of development at which point the system enters operations. Figure 2 is from his original paper and supports this claim [48, 12, 25].

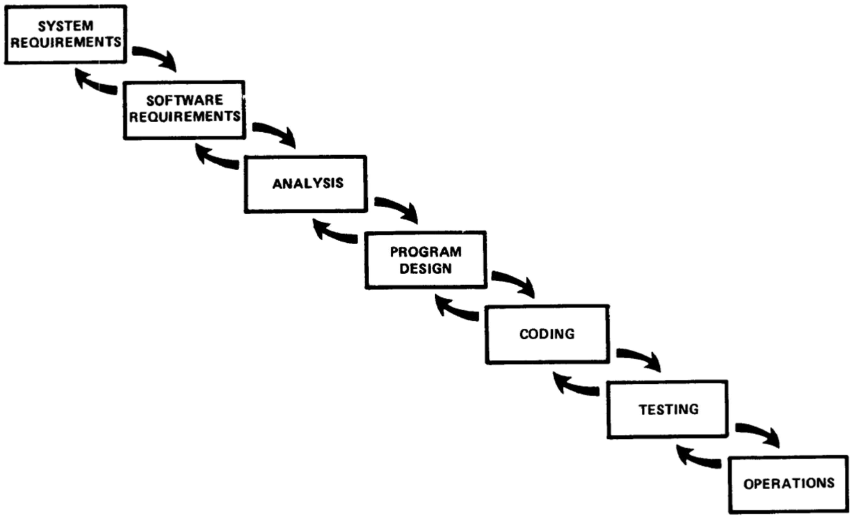


Figure 2. Royce's Waterfall Interaction Diagram [48].

Spiral Development Model

The spiral model is adapted from the waterfall process model, but may include the use of prototypes. It is defined by an iterative risk-driven approach to developing systems. Each iteration through the model will deliver additional capability to the user. An iteration will progress and repeat through stages which define objectives, analyze risks, develop specified capabilities, and develop plans for subsequent cycles. Figure 3 is an example of a visual depiction of progress through the model [25].

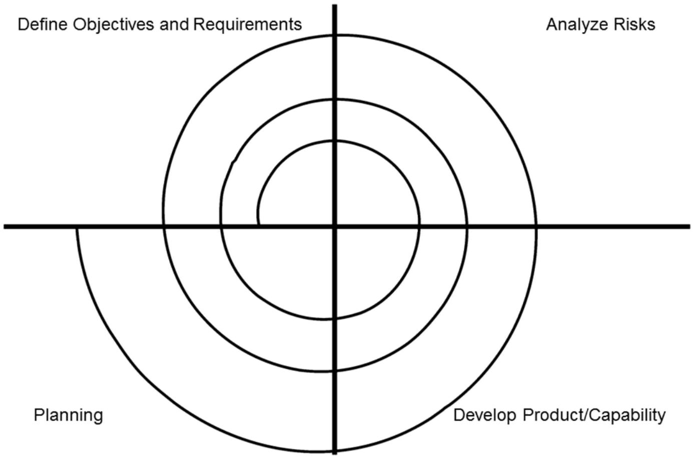

Define Objectives and Requirements
Analyze Risks
Planning
Develop Product/Capability

Figure 3. Spiral Development Model.

Incremental Development Model

In contrast to the waterfall model, incremental development enables some system capability to be delivered before the full system is completed. However, the phases of an incremental approach are typically similar to those found in the water fall model. The incremental model may include the use of prototyping. Additionally, use cases and scenario-based parameterization can be used to manage and schedule development of system increments. Gomaa warns that software architecture, e.g. interfaces and quality attributes, must be carefully designed into the system from the onset because each iteration "forms the basis" of the final deliverable. Thus, early mistakes or poor design decisions can be propagated into finished products. Figure 4 is a graphical depiction of an incremental development model [12].

Rational Unified Process (RUP) Model

The RUP is also iterative model and introduces the concepts of artifacts, workflows, phases and milestones. Artifacts are produced by workflows which are synonymous with processes. Examples of workflows (or disciplines) include requirements definition, analysis, design, and testing. The period of time between the completions of two milestones define phases. Phases are typically named Inception, Elaboration, Construction, and Transition and correspond on a smaller scale to similarly named phases in a system life cycle. Finally, milestones are defined by the achievement of specific objectives and the decision process to move to the next phase [12]. Figure 5 depicts an example of these concepts as well as the work load relationship between workflows and phases.

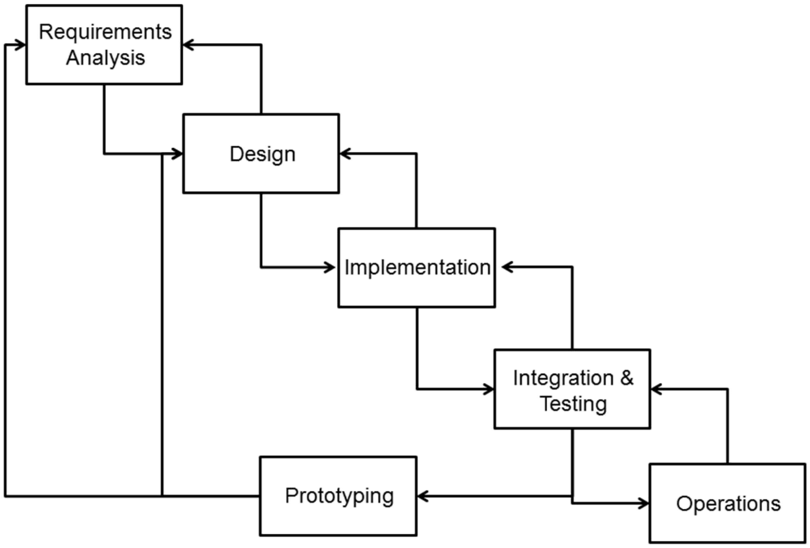


Figure 4. Incremental Development Model.

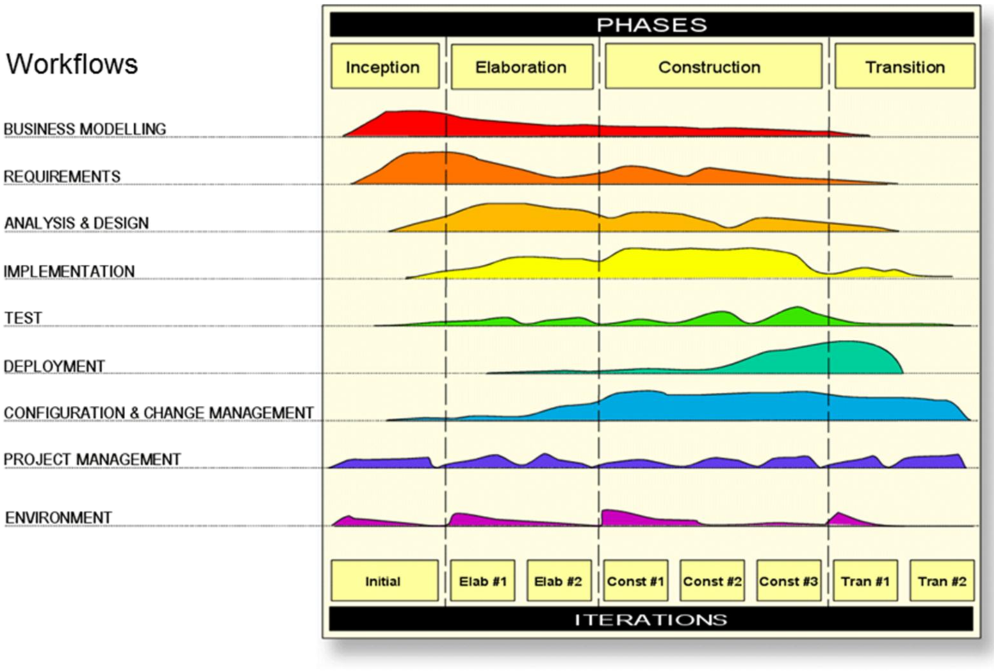


Figure 5. Rational Unified Process Model [49].

"Vee" Model

The Systems Engineering "Vee," as shown in Figure 6, is another development process model that is currently used to develop large systems. The model manages user needs identified during program initiation, i.e., the top left of the diagram. Requirements are developed and decomposed into a detailed design. As the design is implemented and subsystem integrated, the program's requirements are incrementally verified. As the solution is realized, i.e., further integrated into the final system, it will be validated against the original capability need or top-level requirements [25, 41, 33].

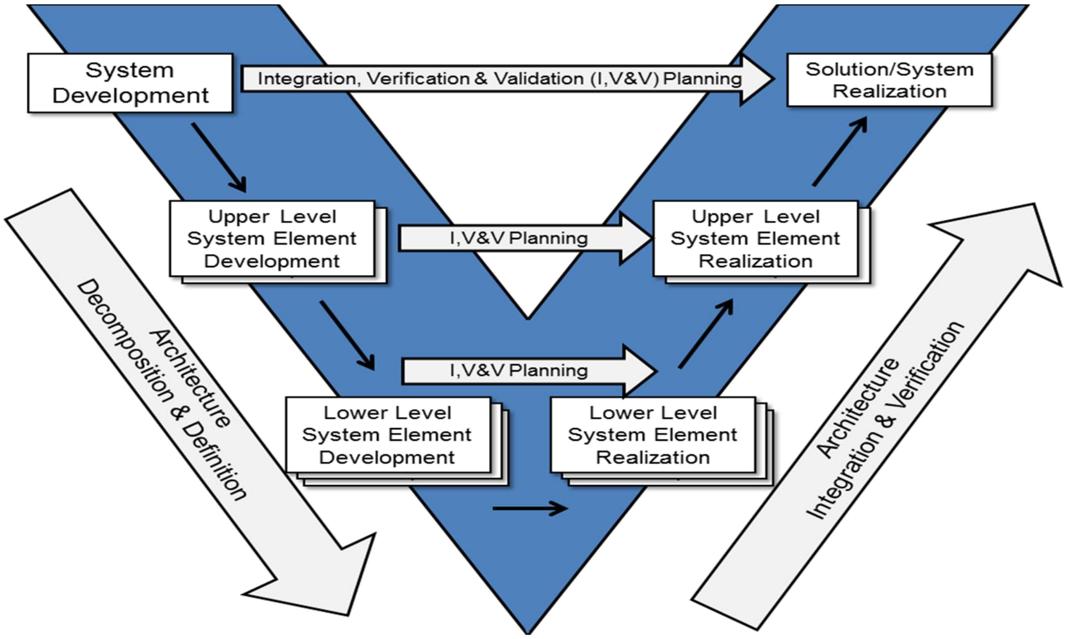

System Development
Integration, Verification & Validation (I,V&V) Planning
Solution/System Realization
Upper Level System Element Development
I,V&V Planning
Upper Level System Element Realization
I,V&V Planning
Lower Level System Element Development
Lower Level System Element Realization
Architecture Decomposition & Definition
Architecture Integration & Verification

Figure 6. Systems Engineering Vee Model [33].

The Defense Acquisition University defines a SE process model that is composed of both technical and technical management processes as depicted in Figure 77. The DoD SE process model identifies eight technical processes and eight technical management processes supporting the total lifecycle management of DoD systems from the initial identification of an operational need to the delivered capability, and disposal [41].

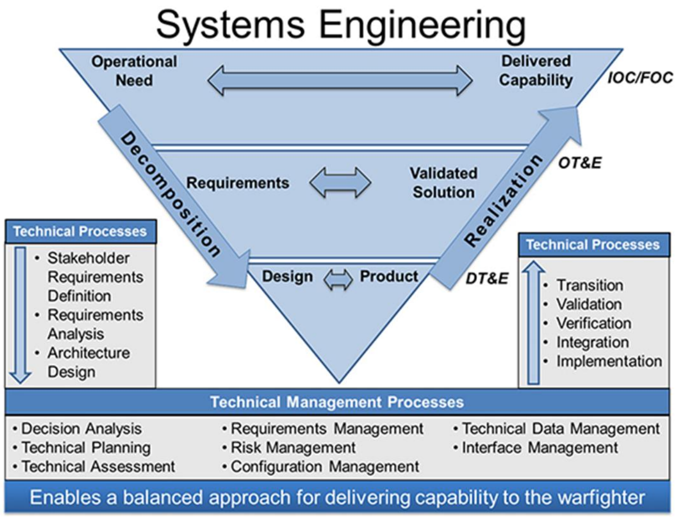


Figure 7. DoD SE Process Model (2014) [41].

Model-Based Systems Engineering

Now that fundamental definitions, concepts and processes of SE have been discussed, the discussion will expand to MBSE. MBSE will build upon the SE ideas and modeling concepts to them as an approach to developing systems. Many of these principles will directly influence answers to research questions or shape the methodology described in Chapter III. This section will discuss key definitions, processes and methods, and tools related to MBSE.

Key Definitions

MBSE, like SE, includes terminology that must be defined to facilitate understanding. These terms include MBSE itself, methodology, process, method, tools, modeling and simulation. Additionally three software engineering terms, i.e., framework, toolkit/toolbox, and architecture have been included in this section to prepare the reader for later discussion.

MBSE

Model-Based Systems Engineering is an emerging methodology which integrates the use of models and modeling principles into various processes encompassed by the field of SE [50]. MBSE provides an organized, repeatable, iterative, and convergent approach to understand, communicate and meet, user needs throughout the development of complex, interdisciplinary systems [51, 52]. More specifically, MBSE is a model-centric approach that supports requirements definition, system design, analysis, and V&V activities whose main deliverables are a model and supporting artifacts which facilitate shared understanding and clear communication amongst stakeholders [53].

More specifically, INCOSE offers the following definition of MBSE: "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [51]. Additionally, Friedenthal *et al.* uses the INCOSE definition and argues that MBSE can enhance communication, reduce risk, improve quality, and increase productivity [54]. These qualities, reduced risk, improved quality and increased productivity, have the potential effect to reduce total life cycle cost [36]. As an emerging methodology, MBSE is being investigated as an effective tool for understanding increasingly complex systems [55, 56, 57, 58]. Furthermore, MBSE, like SE, is a requirements-driven approach. Thus, the first step of an MBSE or SE approach begins with capturing stakeholder needs. Often these stakeholder needs are captured in a concept of operations (CONOPS) [59, 60, 61, 33]. CONOPS typically become the starting point for generating the initial set of system requirements and the foundation for establishing traceability. For these reasons, MBSE is well suited to study QKD systems.

Methodology, Methods, Processes, Tools

A survey of MBSE Methodologies in 2008 defined the distinction between methodologies, processes, methods, and tools. A Methodology was defined as an assemblage of related processes, methods, and tools that can be applied to a common class of problems. A process, it stated, was a sequence of tasks that are accomplished to meet an objective and defined what is to be done but not how. In contrast, a method defines the how and describes the techniques, practices, or procedures for accomplishing a task. Lastly, the study defined tools as a mechanism that can be applied to a method to improve the task's efficiency [53]. These distinctions provide the foundation for understanding these terms throughout this thesis.

Modeling vs. Simulation

It is important to discuss the difference between models and simulations to establish a common vocabulary and the understanding of the nuances of each term. Often the two terms are used together and when used synonymously, are used incorrectly. The IEEE Standard 1233-1996 defines a model as a "representation of a real world process, device, or concept" [62]. Similarly, Banks *et al.* defined a model as a "representation of a system for the purpose of studying that system" [16, p. 15]. Summarily, a model is a "representation or abstraction of something" [44]. All three definitions describe a model as a "representation" of an item of interest, but not, in fact, the item itself.

A key principle or concern is that of a model or an experiment's objective and purpose. Furthermore, both Banks *et al.* and Law and Kelton identify project objectives among the first considerations when defining a model [16, 42]. Similarly, the DoD's GUIDEx identifies "Fitness for purpose" as the number one issue of concern to defense decision makers who seek to use experimentation to solve force capability problems [7]. Arguably, the notion of a "fit-for-purpose" objective (or design) is inherent to the definition of the model itself. That is, a model should be built with a specific purpose in mind at the time of its design.

Banks *et al.* define a simulation as "the imitation of the operation of a real-world process or system over time" [16]. Having already defined a model as a 'representation of an item of interest,' one may be lead to wrongly believe the two are in fact the same. The distinction is as follows: a simulation is always a model, but a model is not always a simulation. Law and Kelton define a simulation as "numerically exercising the model for the inputs in question to see how they affect the output measure of performance" [42, p. 5]. A simulation is a model because it represents a real world item of interest and its behavior, i.e., output, in terms of defined measures of performance.

Simulations may also be described as static, dynamic, deterministic, stochastic, continuous or discrete. A static simulation is a model of a system at a specific time or a model in which the system is not affect by time. Dynamic simulations include the model behaviors as they change over time. A deterministic simulation is a simulation that does not contain any random component and whose output behavior is unique, but known, for a given set of input parameters. Stochastic simulations contain one or more elements that are probabilistic. Furthermore, continuous simulations model behavior or state changes that occur without interruption, i.e. continuously. In contrast, simulations in which the state of the system only changes at particular events are known as discrete [16, 42].

Framework, Toolkit/Toolbox, and Architecture

Lastly, there are four terms that author encountered during research that were often used synonymously in the context of software and simulation: framework, toolkit (or toolbox), and architecture. These definitions will provide the some context in which to understand the relationship of these terms to developing a software simulation. In an effort to provide some clarity to the reader, the following working definitions are provided:

1. Framework: “a set of source code or libraries which provide functionality common to a whole class of applications. While one library will usually provide one specific piece of functionality, frameworks will offer a broader range” of functionality [5].
2. Toolkit (or toolbox): a set of related software components in which components may be modified, used together, or connected to each other to produce other components, functions, or behaviors.
3. Architecture: “high-level design and execution structure of a software system. A software's architecture is analogous to a building's architectural choices for a foundation, frames, window locations, etc” [1].

Processes and Methods

This section will discuss MBSE processes and methods studied and expected to support design and implementation of a QKD system model. First, a discussion of the modeling process is provided and is followed by a review of experimental design. Next, a cursory summary of key software engineering concepts is reviewed. Variance reduction in computer simulations is discussed next. Afterward, the Department of Defense Architecture Framework (DoDAF) is briefly examined. This section concludes with a review of use case modeling.

Modeling Process

Law and Kelton offer six ways to study a system of interest. Figure 8 illustrates the methodology they defined. The first method discussed is to experiment with the actual system. However, cost and practical limitations, such as the lack of existence of the system, may severely restrict the study. Hence the discussion moved to creating a representation of the system for study, i.e., modeling. Physical or iconic models include tangible elements, such as life-sized or scaled components, that can be used to conduct the study and collect data. Mathematical models compose the majority of models built for study. Such models represent the system of interest in terms of "logical and quantitative relationships" [42, p. 5] that can be configured to study the effects of the changes. Some systems, they argue, are too complex to be studied with a closed-form equation or relationship. For these systems, simulation becomes necessary to observe and study the effects of the inputs on the system measure of performance. These performance measurements enable researchers to make comparisons among potential or realized alternatives systems under consideration [42].

Banks *et al.* define a 12-step paradigm for conducting simulation studies that includes problem formulation, planning, verification & validation, experimental design, implementation and documentation as shown in Figure 9. Problem formulation is a critical and preliminary step in the Modeling and Simulation (M&S) development process. Once the problem of interest has been identified and clearly defined, it is appropriate to set objectives and develop a project plan. The objectives identify the research questions to be answered by the M&S project. The project plan is analogous to a research methodology and should include the identification of resources where appropriate. Model conceptualization involves the process of identifying, abstracting, and decomposing the behaviors of the system to be studied. Data collection facilitates the parameterization of the inputs, outputs, and behaviors to be translated into the model. Model translation is the process of implementing the concept model and parameters into the model or simulation itself; it often involves the use of a computer program. Verification is the process of confirming that model is built correctly whereas validation is the process of ensuring the correct model was constructed. Experimental design shapes which parameters or configuration will be studied and the how the statistical rigor is applied (experimental design will be discussed in a later section). Production runs and analysis are the processes of executing the model, collecting, and assessing the output against performance measures. Documentation enables confidence in the operation of the model by providing a means of understanding its design and implementation. Additionally documentation provides a means of evaluating progress towards model objectives. Finally, implementation involves use of the model to answer or solve the problem [16].

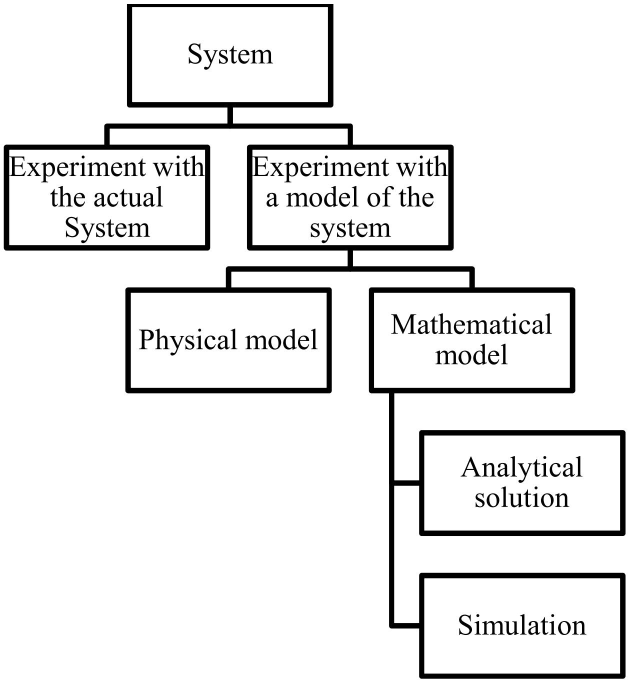

System
Experiment with the actual System
Experiment with a model of the system
Physical model
Mathematical model
Analytical solution
Simulation

Figure 8. Ways to study a system [42].

This process is strikingly similar to the 10-step process Law and Kelton defined on the same subject. In their step 1, the problem is formulated and the project is planned (a combination of Banks *et al.*'s steps 1 & 2). In step 2, data is collected and a model is defined (a combination of Banks *et al.'s* steps 1 & 2). Next the conceptual model is validated which is followed by implementation in a computer program and verification (a combination of Bank *et al.'s* steps 5-7). Law and Kelton's fifth step is to perform pilot runs. Step 6 is another validation steps that compares the results of the pilot runs with the actual system or expectations. The next two steps in the process are to design one or more experiments and then conduct production runs. The ninth step to analyze the data. The process culminates in documentation and results presentation [42]. Figure 10 shows the process depicted in a flow chart.

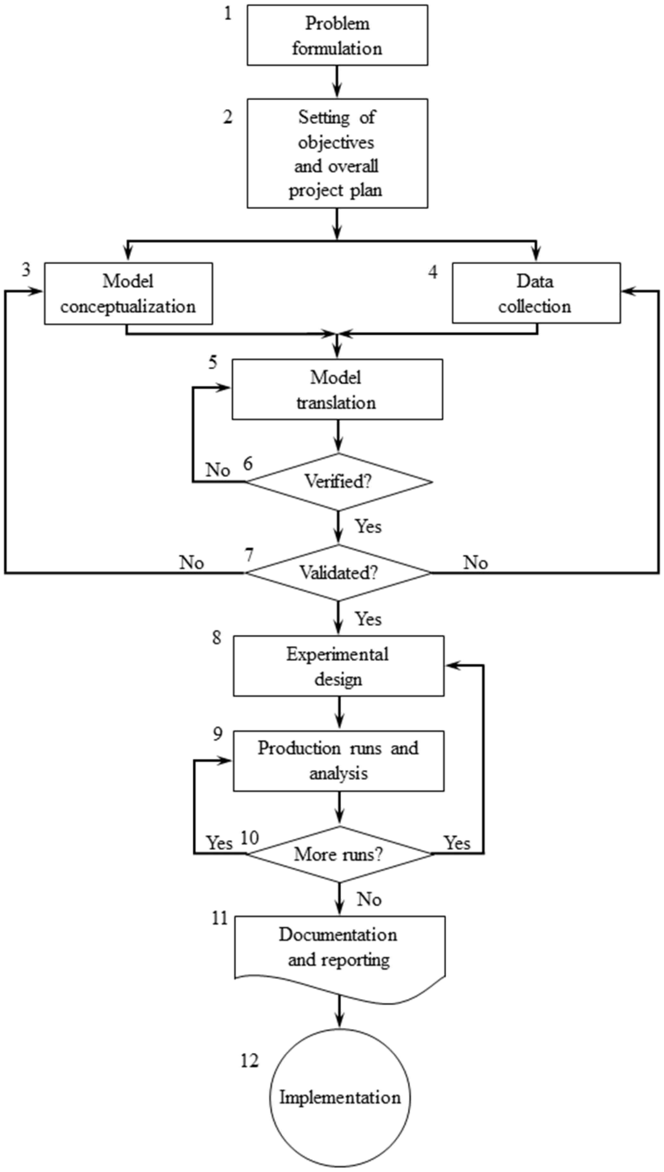


Figure 9. 12 Simulation Steps [16].

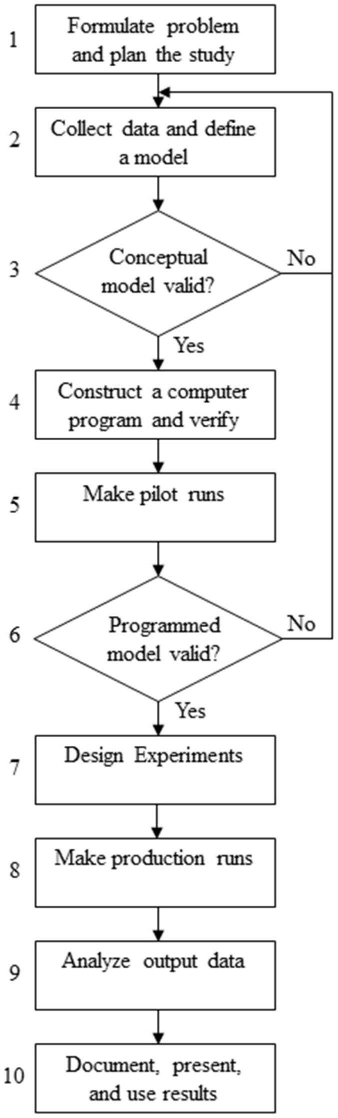

1
Formulate problem and plan the study
2
Collect data and define a model
3
Conceptual model valid?
No
Yes
4
Construct a computer program and verify
5
Make pilot runs
6
Programmed model valid?
No
Yes
7
Design Experiments
8
Make production runs
9
Analyze output data
10
Document, present, and use results

Figure 10. 10 Simulation Steps [42].

From amongst these two paradigms, the Bank *et al.* version was selected by the researcher because of the additional delineation of the processes. The researcher believed that additional steps added clarity to the process. For example, the validation loop in step 7 provided a clear opportunity to return to either data collection or model conceptualization. Additionally he believed the Banks *et al.* paradigm captured the process more accurately captured the process he intended to follow. Ultimately one could argue that selection was subjective.

Experimental Design

According to the GUIDEx, an experiment is "the manipulation of objects under controlled conditions while taking precise measurements" [11, p. 7]. The GUIDEx defines an experimental process that is consistent and complementary to the guidance offered in the aforementioned modeling discussion and tailored toward defense experiment planners. The relationship between models and experiments can be further observed in the GUIDEx's discussion of experiment development in which model development is a key element. Figure 11 is the GUIDEx flowchart which summarizes the steps deemed necessary in defense-based experiments [16, 7].

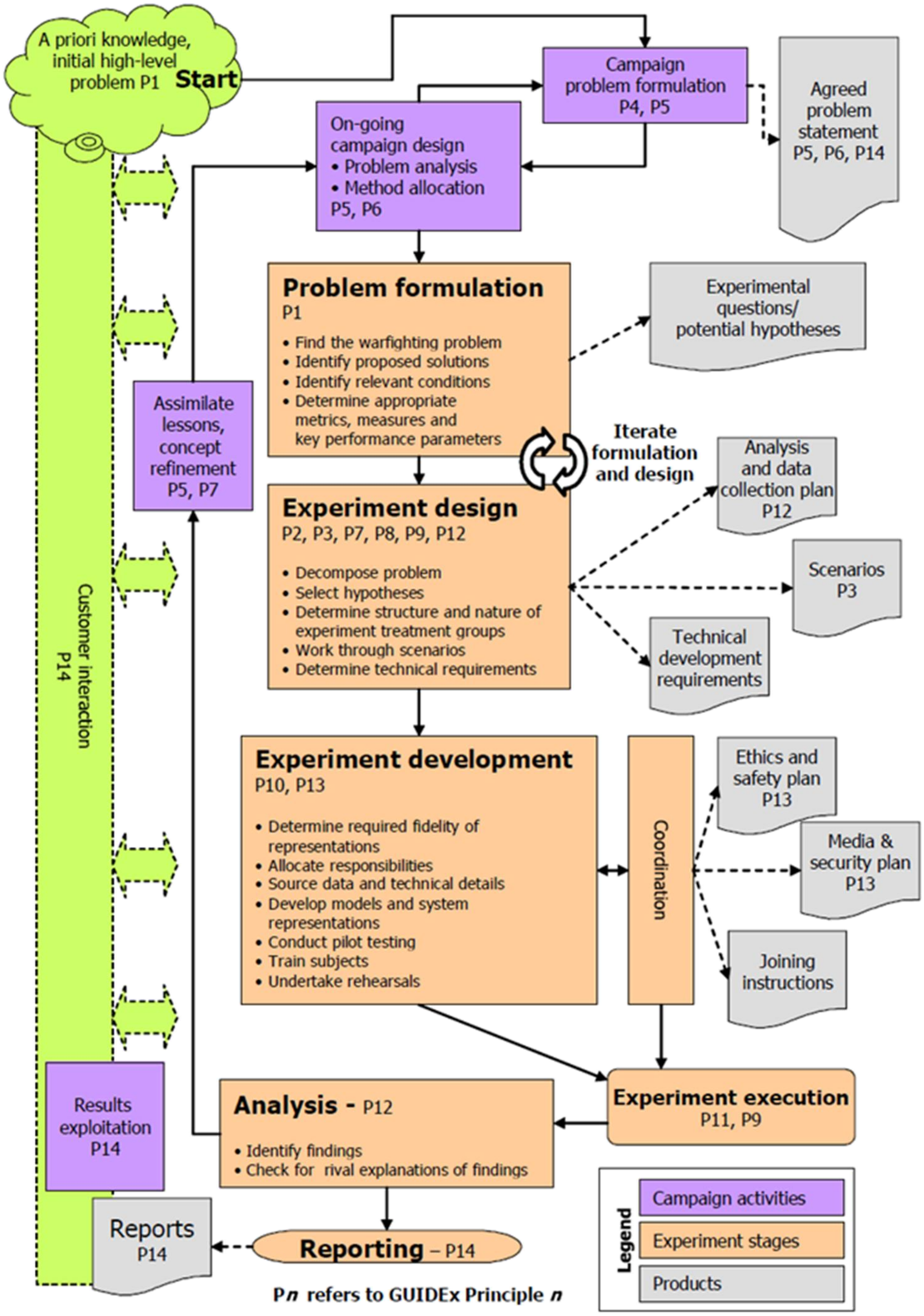

A priori knowledge, initial high-level problem P1
Start
Campaign problem formulation P4, P5
Agreed problem statement P5, P6, P14
On-going campaign design
• Problem analysis
• Method allocation
P5, P6
Problem formulation
P1
• Find the warfighting problem
• Identify proposed solutions
• Identify relevant conditions
• Determine appropriate metrics, measures and key performance parameters
Experimental questions/ potential hypotheses
Assimilate lessons, concept refinement P5, P7
Iterate formulation and design
Analysis and data collection plan P12
Experiment design
P2, P3, P7, P8, P9, P12
• Decompose problem
• Select hypotheses
• Determine structure and nature of experiment treatment groups
• Work through scenarios
• Determine technical requirements
Scenarios P3
Technical development requirements
Customer interaction P14
Experiment development
P10, P13
• Determine required fidelity of representations
• Allocate responsibilities
• Source data and technical details
• Develop models and system representations
• Conduct pilot testing
• Train subjects
• Undertake rehearsals
Coordination
Ethics and safety plan P13
Media & security plan P13
Joining instructions
Experiment execution P11, P9
Results exploitation P14
Analysis - P12
• Identify findings
• Check for rival explanations of findings
Reports P14
Reporting – P14
Pn refers to GUIDEx Principle n
Legend
Campaign activities
Experiment stages
Products

Figure 11. GUIDEx Flowchart [7].

Montgomery defines an experiment as “A test or series of runs in which purposeful changes are made to the input variables of a process or system so that we may observe and identify the reasons for changes that may be observed in the output response” [13, p. 1]. Experiments often are composed of five elements: 1. a treatment, 2. a (main) effect, 3. the experimental or sampling unit, 4. a trial, and 5. the analysis [7, 11, 3]. *Treatments* are the "purposeful changes made to the input variable" [3] administered by the experimenter or researcher on the experimental unit [13, 3]. The *main effect* is observed as the results of a trial or in other terms, it is the average change in the response [11, 42]. The *experimental unit* is the item being studied [3, 4]. A *trial* or *run* is "one observation of the experimental unit under treatment" [11, p. 8]. The *analysis* is the phase in which results are evaluated and compared. In many cases results are compared between runs, where analysis is done on the *response* or output of the experiment [11]. The response is also referred to as the dependent variable [13, 3].

Three basic principles that shape experimental design are randomization, replication, and blocking. Randomization in this context has two meanings. The first meaning indicates that experimental material will be allocated by treatment in a randomized manner, and the second denotes that order of the test runs will be scheduled randomly. Replication refers to independent repeated runs of the factor combinations. In computer simulations, this principle is employed by the use of different seeds input to the random number generator(s). Blocking is a method in which the experimental units are organized in homogeneous clusters. Organizing them in this manner and randomly allocating treatments within each block is an attempt to mitigate the effects of nuisance factors. Nuisance factors are factors in which the experimenter is not interested but which may affect the response variables [13, 3].

Rigorous experiments based on statistical principles often involve the use of a hypothesis test, with a null and an alternative hypothesis. The objective of the experiment will often attempt to disprove the null hypothesis and therefore conclude that alternative hypothesis should be accepted [4, 13]. For example, if the null hypothesis is that yield of the two-photon signal state is equal to the yield of the two-photon decoy state, then the alternative hypothesis might be that two yields are not equal. Thus if the statistical analysis of the experimental results infer that null hypothesis should be rejected, then the experimenter would recommend acceptance of the alternative hypothesis.

Software Engineering and Design

Software engineering has been defined by Mills as “the systematic design and development of software products and the management of the software process” [63, p. 414] He further explains that software engineering can be an integral step in systems engineering. Similarly, Gomaa defined software design method as “a systematic approach that describes the sequence of steps to follow in order to create a design, given the software requirements of the application” [12, p. 5]. These concepts and associated terminology are relevant to facilitate the understanding of the MBSE discussion.

Two dominant paradigms shape the design approach to developing and implementing software solutions: a functional paradigm and an object-oriented paradigm. In the functional approach, algorithms and system behaviors are decomposed into functions which are then implemented to control the flow of the system [64]. Functional decomposition is a term that is used in both paradigms of software design as well as in SE. Alternatively, this process may be described as functional analysis, functional breakdown, analysis modeling, or process decomposition [25, 12, 65]. The object-oriented approach builds on the concept of functional decomposition of behaviors and adds a data-centric view of the design trade space.

In the object-oriented paradigm, the concepts of classes, inheritance, and information hiding make this design approach unique from the functional paradigm. A class is defined as the implemented form of a “data type that is defined by operations that manipulate it” [12, pp. 523-524]. In the functional paradigm data and operations are not packaged together in this way. In fact, only data can be packaged together in structures. Information hiding is characterized by the intention to prevent unnecessary access data. This concept is designed to overcome some of the risk associated with the function design approach. Finally, the concept of inheritance enables the replication or extension of class behaviors to other classes. In functional paradigm, there are no classes to extend [12, 64].

Another noteworthy pair of concepts involves the communication patterns between software objects in tasks that involve timing or sequencing constraints. Synchronous and asynchronous communication patterns shape the characteristics of such systems that rely on timing. In a synchronous communication pattern, an object will send a message to another object and await a response before proceeding to the next task. In contrast, a design that employs an asynchronous communication pattern would not require that first object wait for the response before proceeding to its next task [12].

Variance Reduction in Computer Simulation

In order to make statistically relevant inferences among alternative designs, it is desirable to reduce the total variance. The total variance of the difference of the means of the response variables (outputs) can be reduced by inducing increased covariance. If the same random numbers are used between each set of simulation runs, the means of the response variables will become correlated and the total variance can be reduced. This method of using common random numbers in simulations is known as *correlated sampling* and is a variance reduction technique that is relatively easy to perform via computer simulation [16, 42].

Architecture Framework

The DoD has developed its own architecture framework, known as DoDAF, to assist its managers with their decision making processes. The DoDAF provides customizable viewpoints that can be used to facilitate the total lifecycle management of complex systems. Additionally, the DoD specifically acknowledges the "Fit-for-Purpose" nature of architectural content that is used to populate DoDAF viewpoint products [27]. This "Fit-for-Purpose" concept is consistent with a key principle of model design and integrates well with MBSE as the DoDAF products are models themselves, i.e., representations of items of interest.

Use case modeling

Use cases are employed to describe functional requirements in terms of interactions between users and the system of interest; they are intended to describe what actions the system performs rather than how the system does it. Actions are initiated by inputs from an actor to the system. Actors are typically defined as entities external to the system and represented in UML by a stick figure because of the high frequency in which actors are human. However, it is possible for non-human entities, such as a monitoring sensor or a hardware input/output device, to be modeled as an actor in a use case [12].

A use case's main sequence should describe the expected sequence of events between the system and its actors. Therefore, MBSE can effectively employ use cases to "define a sequence of interactions between one or more actors and the system" [12, p. 535]. A use case template outlined by Gomaa, consistent with the RUP Use Case template, provided the basis for the use cases defined in this document [12, 66, 67].

Tools

Many tools exist to improve the efficiency of MBSE method application. In fact, a pencil and paper meet the definition of tool identified by the INCOSE-sponsored survey [53]. However only the tools that were somewhat unique to MBSE or directly applied to this study will be discussed in this section; thus the discussion is not comprehensive. The tools are grouped in this discussion by whether they primarily support design or implementation, i.e. belong on the left or right side of the Vee Model. Within both categories of tools, one can find a series of open source and commercial products.

Design Tools

A common set of SE modeling tools are methods which define the characteristics and style of the artifacts created to capture design. The Unified Modeling Language (UML), Systems Modeling Language (SysML), Integrated DEFinition (IDEF0), and the Structured Analysis Design Technique (SADT) are common approaches. SysML is an SE adaptation of UML whose origins began with software development. UML and SysML tend to focus on application and data structure, processes, behaviors, and architecture [68, 69]. Likewise, IDEF0, commissioned by the US Air Force to analyze and communicate system functional characteristics, has origins with SADT which is a process-oriented approach [53, 70].

Additionally, Microsoft Office provides a flexible set of products in the form of Word, PowerPoint, Excel, Access, Visio, and Project which can be used to support MBSE. However these tools are not strictly designed for modeling or SE and therefore do not provide specific solutions to task efficiency. For example, both Word and Access can be used to track requirements, but neither is inherently designed to do so and must be configured by the user. Likewise, PowerPoint and Visio offer a means to create modeling diagrams, but require some configuration by the user to do so. Project is perhaps the modeling-friendly as it supports schedule and resource management out of the box.

In contrast to the Microsoft Office Suite of tools, Vitech's CORE and Sparx System's Enterprise Architect are designed to support SE. These tools provide integrated database support by project, i.e. file to facilitate requirements and overall system design decision traceability. Both tools are designed to support requirements definition, use case modeling, architecture design, and software and database design among other capabilities.

An open source design tool, called UMLet, is a free option with the capability to create class, object, sequence, and use case diagrams. UMLet does not support an integrated database which causes each diagram to be independent from all others. However, it does create output in an XML-based format which makes it fairly versatile. Additionally UMLet works as a stand-alone product or can be integrated with some implementation tools.

Implementation Tools

The Eclipse Integrated Development Environment (IDE) is an open source IDE that supports software project development in Java, C, C++, PHP, and other languages. The Eclipse IDE supports version control by providing integration with code repository hosts such as GitHub. Additionally Eclipse has been adapted to work with an open source simulation framework known as OMNeT++ [71].

OMNeT++ is a free discrete event simulator that is based on C++ and designed for building network simulations. As a framework, OMNeT++ is extensible, modular, flexible, and provides a series of library functions to model network and network functions. OMNeT++ supports development of both wired and wireless communication networks, sensor networks, ad-hoc wireless networks, Internet Protocols, performance modeling, and simulation capabilities [72].

Quantum Key Distribution

Quantum Key Distribution (QKD) is the most mature application of the quantum information field and is touted as a revolutionary technology which offers the capability for two authenticated parties, Alice and Bob, to generate a symmetric secret key for use in a cryptosystem. The nature of quantum mechanics enables the unique ability of a QKD system to detect the presence of an eavesdropper attempting to subvert the secure distribution of key material. This section provides an overview of fundamental cryptographic concepts, the first QKD protocol, QKD vulnerabilities, the decoy state protocol, and concludes with a description of the qkdX framework.

Cryptography

Cryptography involves the study and application of methods used to provide secure communications in the “presence of adversaries” [73]. Cryptography is implemented through a cryptosystem consisting of an encryption algorithm and at least one key. Symmetric encryption algorithms such as the one-time-pad (OTP), Data Encryption Standard (DES), Triple DES (3DES), Advanced Encryption Standard (AES), Blowfish, Rivest Cipher 4 (RC4), and RC5 require only one key and can be employed using a shared key generated by QKD systems [74, 32, 10, 30].

In a symmetric key system, plaintext, m, is transformed by the encryption algorithm, E, using the encryption key, K, into ciphertext, $E_K(m)$, which is computationally difficult to decode back to plaintext by someone, i.e., an adversary, who does not possess the key [31, 74]. For a person with the symmetric key, a decryption algorithm, D, and key, K, are used to transform the ciphertext, i.e., $D_K(E_K(m))$, back to the plaintext, m. Figure 12 illustrates this process in a block diagram [10].

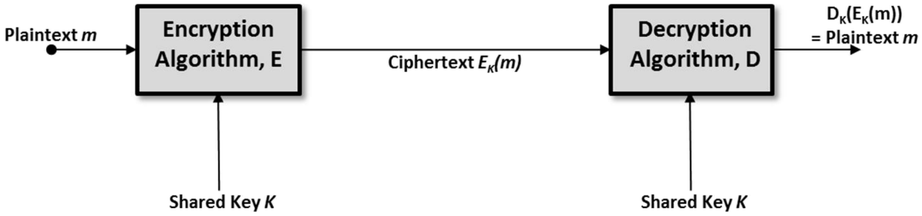


Figure 12. Symmetric Key Cryptosystem Block Diagram.

The First QKD Protocol: BB84

The origins of QKD technology can be traced back to Stephen Wiesner, who first developed the idea of fraud-proof quantum money [75]. Later, Charles Bennett and Gilles Brassard operationalized this concept when they proposed the first QKD protocol (i.e., BB84) where single photons, representing quantum bits (qubits), are used to securely generate shared cryptographic key [2]. BB84 is a polarization- or phase-based prepare and measure QKD protocol defined by of eight phases of interactions between Alice and Bob as illustrated by Figure 12. First, Alice and Bob must authenticate their identities to each other. Transactional authentication may also be employed upon completion of the protocol for additional security [18]. Next, they perform the quantum exchange until Bob detects enough qubits to begin processing a secret key. The third phase, sifting, occurs when Alice and Bob compare their measurement bases to enable Bob to correctly interpret qubit values. The fourth phase is complete after Alice and Bob estimate the number of errors in this sifted key. In the next phase, they correct the errors in the sifted key. In the sixth phase, Alice and Bob estimate entropy, i.e. the amount of information leaked during the previous phases. Phase seven is known as privacy amplification and attempts to mitigate the losses due to the leaked information. The last phase occurs when Alice and Bob confirm they have matching keys by comparing a hash that each calculates [76].

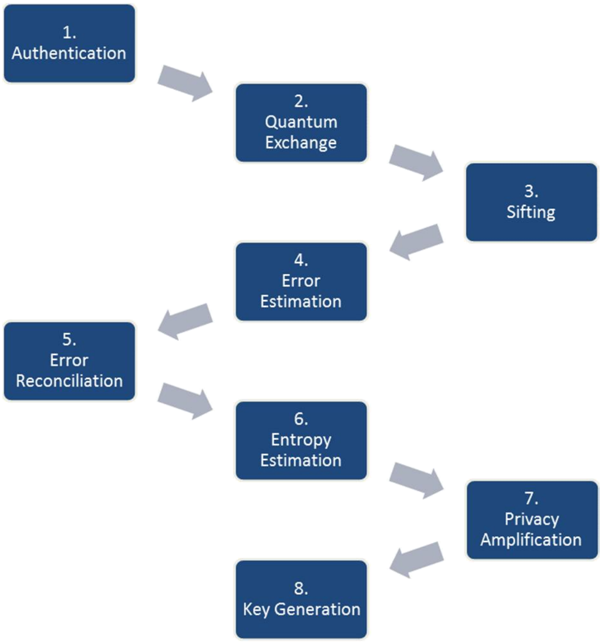


Figure 13. 8 Phases of BB84 [76].

During quantum exchange, one of four polarization states ↔ (horizontal), ↕ (vertical), ⤢ (diagonal), and ⤡ (anti-diagonal) is used to encode and decode qubits transmitted over an optical channel. That is, Alice will a randomly select a bit value (i.e., “0” or “1”) and randomly select a basis (i.e., rectilinear “⊕” for the orthogonal polarization pair ↔, ↕ or diagonal “⊗” for the orthogonal polarization pair ⤡, ⤢). For example, when the bit value “0” and the basis “⊕” are selected, the qubit is prepared in the horizontal polarization state ↔. Similarly, when the bit value “1” and the basis “⊗” are selected, the qubit is prepared in the diagonal polarization state ⤢. In this manner, Alice randomly prepares qubits through polarization modulation and sends them to Bob where he randomly selects a basis (⊕ or ⊗) to measure each qubit. When Alice’s prepared and Bob’s measured bases agree, the qubit is correctly read with a high probability. Otherwise a random result occurs as depicted by the shaded cells in the last column of Table 1 [76]. The preparation, transmission, and measurement of qubits between Alice and Bob are inherently quantum in nature, while the remainder of the BB84 protocol (sifting, error estimation, error reconciliation, and privacy amplification) is entirely classical. During the classical phases, Alice and Bob use traditional information theory practices and communicate over a classical communications channel [26, 18].

Table 1. Polarization-Based Prepare and Measure Example.

The Sender “Alice” Prepares			The Receiver “Bob” Measures	
Random Bit Value	**Random Basis**	**Prepared Polarization State**	**Random Basis**	**Measured Bit Value**
0	⊕	↔	⊕	0
0	⊕	↔	⊗	0 or 1
1	⊕	↕	⊕	1
1	⊕	↕	⊗	0 or 1
0	⊗	⤡	⊕	0 or 1
0	⊗	⤡	⊗	0
1	⊗	⤢	⊕	0 or 1
1	⊗	⤢	⊗	1

The distinctive character of QKD technology makes it well-suited for high-security applications such as military, banking, and government environments, where information protection is of primary interest. The security of QKD is based on the principle of quantum uncertainty inherent in the BB84 protocol where interference on the quantum channel necessarily increases the Quantum Bit Error Rate (QBER) [26, 18]. If an eavesdropper, often known as “Eve,” attempts to listen on the quantum channel, she introduces additional errors and increases the QBER [77, 78]. Since the source of quantum bit errors cannot be determined, all errors are assumed to be attributed to Eve; thus the QBER must be closely monitored. If the QBER threshold is exceeded, the key generation process is generally aborted, as it is assumed an adversary is interfering with the quantum key exchange [18]. However, practical design and implementation limitations can quickly undermine pristine theoretical security proofs [79, 80].

QKD Multi-Photon Vulnerability and the Photon Number Splitting Attack

Proofs and discussion of BB84 and other protocols assume a set of idealities, such as the employment of a perfect single photon source and detector, which are not possible with current technology [79, 80, 81]. As practical on-demand single photon sources are not available, most prepare and measure implementations attenuate a classical laser pulse down from millions of photons to a Mean Photon Number (MPN) on the order of 0.1. These sub-quantum pulses are often referred to as “weak coherent pulses" and are represented probabilistically by the Poisson distribution using MPN as the input parameter. The selection of the MPN is influenced by the losses in the optical channel and assumptions regarding the eavesdropper's technology [26] .

A low MPN results in poor qubit throughput and thus low key generation rates. However, a low MPN is often desirable to reduce the likelihood of multi-photon pulses, each of which exposes information about the distributed cryptographic key to an eavesdropper. This multi-photon security vulnerability has been subject of a number of attacks against the quantum channel. More specifically, the Photon Number Splitting (PNS) attack was conceived to gain maximum information from this vulnerability without detection. The theoretical PNS attack suggests that an all-powerful eavesdropper (Eve) can develop the same secret key without making her presence known to the authorized participants. She does so by blocking all single photons and stealing photons from the multi-photon pulses as they are in transit from the sender (Alice) to the receiver (Bob) [79, 19, 20, 21]. The decoy state protocol was introduced to mitigate the ability Eve to hide her presence during this attack [24].

The Decoy State Protocol

The decoy state protocol represents a significant advancement in QKD technology, as the protocol is relatively cheap and easy to implement against the PNS vulnerability and can be configured to increase throughput on the quantum channel. The protocol was introduced in 2003 [24] and quickly improved in an initial series of works [28, 29, 82] and later [6, 83, 84, 85, 86, 87] by a group of researchers (supplemental works of note include [88, 89, 90, 91, 92]). In this three state configuration, the signal state, μ, facilitates higher key distribution rates and greater operational distances through an increased MPN, while the decoy state,ν, is used to increase the likelihood of detecting an attack using differential statistical analysis. Utilizing different MPNs for the signal and decoy states, allows the QKD system to detect photon number specific interference using the decoy state protocol security condition (described in the next section). The vacuum state is used to determine the detector error rate at Bob, where erroneous detections, known as "dark counts" are measured [93, 14].

Decoy state enabled QKD systems utilize the conventional signal state plus dedicated security states (decoy and vacuum). The signal states are used to transmit the qubits used for generating the shared secret key. The decoy states are used to increase the likelihood of detecting photon number specific interference, i.e., the PNS attack, on the quantum channel through statistical differentiation with the signal state. The vacuum states are used to determine the detector dark count rate when no photons are present. Alice randomly selects the type of state according to pre-determined occurrence percentages [14, 94].

An outline of the decoy state protocol, decomposed into multiple steps for clarity, is described in [24, 28, 14]. The first step in the protocol is known as quantum exchange, where signal, decoy, and vacuum states are randomly transmitted across the quantum channel according to their unique MPN and occurrence percentage. Each pulse must have identical characteristics (e.g., wavelength, duration, etc.) other than the MPN, such that Eve cannot distinguish a decoy state from a signal or vacuum state.

Table 2. Decoy State Protocol.

Step	Name	Detailed Description
1	Quantum Exchange	Alice randomly encodes and transmits signal, decoy, and vacuum states (i.e., qubits) while Bob measures each qubit. This step is typically described as "quantum exchange."
2	Sifting	Bob announces the basis he used to measure each qubit. Alice responds by announcing which bases are correct, along with the state: signal, decoy, or vacuum for each pulse sent. Alice and Bob perform sifting on both the signal and decoy state qubits for matched bases.
3	Count Decoy State Errors	For the sifted decoy state qubits, Alice announces the encoded bit values to Bob over the classical channel. Bob performs a comparison of Alice's prepared bits and his measured values, and records the number of errors for use in step 5.
4	Signal State Error Reconciliation	For the sifted signal state qubits, Alice and Bob perform error reconciliation where quantum communication errors are counted and corrected for using bi-directional error correction algorithms over the classical channel. The number of corrected errors is recorded for use in step 5.
5	Calculate Gains, QBERs, and Dark Counts	The signal and decoy gains Q_μ and Q_ν, respectively are calculated from the measured number of sifted qubits and number of pulses sent. The signal QBER E_μ is calculated from the measured number of errors in the reconciled key and the number of sifted signal states, while the decoy state QBER E_ν is calculated from the measured number of errors in the sifted decoy state qubits. The number of detections measured by Bob when Alice sends vacuum states is used to determine the system's dark count probability,Y_0.
6	Perform Decoy State Security Check	A comparison of signal and decoy estimated photon number specific yields Y_n and QBERs e_n is performed. If they are not the same (i.e., within a predetermined tolerance), an eavesdropper is assumed to be listening on the quantum channel and the key is considered compromised.
NOTE: The decoy state protocol may be described differently based on implementation decisions. For example, in a typical communications network, i.e., TCP/IP, configuration, it may be more efficient for Alice to combine steps two and three by announcing the decoy bit values when she announces the prepared bases.		

After quantum exchange, Alice and Bob perform sifting where they announce the bases used to prepare and measure qubits along with each qubit's state (signal, decoy, or vacuum). For the signal and decoy states non-matching bases are removed from the Bob's key buffer. In contrast, the correctly matched signal states contribute to a sifted secret key and the decoy states are used for security analysis. Sifting is not necessary on vacuum states, since all vacuum detections are dark counts regardless of the basis [28, 14].

Steps 3 and 4 are responsible for identifying, counting, and correcting errors in the signal and decoy state transmission. Counting of decoy state errors is accomplished by publically announcing and comparing the prepared and measured bit values, while error reconciliation techniques such as winnow, cascade, or Low Density Parity Check (LDPC) are used to perform bi-directional error correction of Alice's and Bob's sifted signal qubits without revealing specific bit values [26].

In step 5, a number of decoy state protocol unique calculations are made from performance measurements: the signal gain Q_μ is the ratio of Bob's sifted signal detections to Alice's number of signals pulses sent; the decoy gain Q_ν is the ratio of Bob's decoy sifted detections to Alice's number of decoy pulses sent; the signal QBER E_μ is the ratio of Bob's post-sifted signal errors to the number of sifted signal pulses; the decoy QBER E_ν is the ratio of Bob's post-sifted decoy errors to the number of sifted decoy pulses; and the dark count yield Y_0 is the ratio of Bob's erroneous detections to Alice's total number of vacuum pulses sent [24, 6].

It is important to note that not all research groups agreed on the definition of gain. For example, Lo *et al.* and Ma *et al.* described gain as the probability of detection events at Bob given that a pulse was sent by Alice [28, 14]. In contrast, Zhao *et al.* define gain in terms of the ratio of the number of sifted detections to the total number of qubits or pulses sent [6]. Specifically, the difference is whether or not gain is defined in terms of sifted detections or total detections. The selection of sifted or total detections appears to depend on each group's interpretation of Gottesman *et al.'s* secret key rate equation with respect to the overall protocol efficiency, i.e. whether or not the equation accounts for sifting [77, 14]. QKD implementations must be careful to identify which definition of gain is used.

Step 6 is responsible for executing the decoy state security condition which ensures there is no unauthorized interference on the quantum channel. The security check is conducted through differential statistical analysis of the decoy state protocol security condition.

The Decoy State Protocol Security Condition

The decoy state protocol security condition is described as:

$$Y_n = Y_n^{signal} = Y_n^{decoy} \qquad \text{Decoy State Security Condition for Yields (1)}$$

$$e_n = e_n^{signal} = e_n^{decoy} \qquad \text{Decoy State Security Condition for Errors (2)}$$

where the photon number dependent yield Y_n is defined as the conditional probability that Bob detects an optical pulse given that Alice sent an n-photon pulse and the photon number dependent QBER e_n is defined as the conditional probability an error occurred given that Bob detects the n-photon pulse from Alice [28, 14].

The security condition asserts the yields Y_n and QBERs e_n for the signal and decoy states should always be the same (i.e., within prescribed tolerances) for each n-photon pulse. This is because the quantum transmission efficiency and error rate are fixed for the quantum channel. For example, when no interference is present $Y_1^{signal} = Y_1^{decoy}$ should always be true because the end-to-end transmission efficiency does not change; thus the likelihood of any pulse with one photon propagating from Alice to Bob is identical, regardless of state. Likewise, the QBERs e_n are based on device and alignment imperfections over the quantum channel and should always be the same for each photon number. Thus Y_n and e_n vary only with respect to the photon number per pulse [28, 14].

In practical QKD implementations, the photon number dependent yields and QBERs are strictly estimates of expected values. They are estimates because QKD realizations often use Avalanche

Photodiode Detectors (APDs), which are threshold detectors that cannot determine the specific number of photons received in each optical pulse. Therefore, Bob cannot precisely measure the n-photon yields Y_n or QBERs e_n; they are estimated from the measured state gains $Q_{\mu}, Q_{v,}$ and QBERs E_{μ}, E_v [28].

Estimating Photon Number Dependent Yields

The photon number depended yield Y_n is defined as [28]:

$$Y_n = Y_0 + \eta_n - Y_0\eta_n \cong Y_0 + \eta_n \qquad \text{Photon number dependent yield (3)}$$

and Y_0 is the dark count (rate):

$$Y_0 = \frac{\text{Number of dark count detections}}{\text{Number of vacuum pulses sent}} \qquad \text{Dark count rate (4)}$$

where η_n is the photon number dependent efficiency, and the joint probability $Y_0\eta_n$ is typically disregarded because it is very small compared to Y_0 and η_n. Treating each photon independently, the photon dependent efficiency η_n is based on the number of photons n in each pulse and the end-to-end efficiency η (commonly expressed as the likelihood of successful propagation or in the negative sense as loss measured in dB):

$$\eta_n = 1 - (1 - \eta)^n \qquad \text{Photon number dependent efficiency (5)}$$

where

$$\eta = \eta_{QuantumChannel}\eta_{Bob}\eta_{detector}. \qquad \text{End-to-end efficiency (6)}$$

The overall efficiency η is the product of the quantum channel efficiency $\eta_{QuantumChannel}$, the receiver's efficiency η_{Bob}, and the efficiency of the Bob's detector $\eta_{detector}$. The quantum channel efficiency is often approximated as $\eta_{QuantumChannel} = 1 - 10^{-\alpha*\ell/10}$ or a measured loss in dB. Bob's efficiency η_{Bob} describes a weak coherent pulse's ability to propagate through Bob's components to his detectors and is typically measured and represented as loss in dB. The detector efficiency $\eta_{detector}$ is device dependent and typically represented as a percentage (e.g., 10%) which defines the detector's probability of detecting a weak coherent pulse which arrives during a gating period [28, 14, 81]. The addition of the protocol efficiency $\eta_{protocol}$ accounts for the signal state occurrence percentage (i.e., < 100%). When studying the decoy state protocol, the protocol efficiency $\eta_{protocol}$ must also be considered:

$$\eta = \eta_{QuantumChannel}\eta_{Bob}\eta_{detector}\eta_{protocol}.$$ End-to-end efficiency with protocol efficiency (7)

In some cases the total efficiency is known and can be used to calculate the yields $Y_n^{signal}, Y_n^{decoy}$; however, assuming the channel is untrusted, the efficiency can be calculated from the measured gains. The relationship between the signal gain Q_μ and the efficiency η is described in Eqs. Signal gain (linear combination of the infinite series) (8) - End-to-end efficiency (in terms of dark count and signal gain) (11)

$$Q_\mu = Y_0 e^{-\mu} + Y_1 e^{-\mu}\mu + \frac{Y_2 e^{-\mu}\mu^2}{2} + \cdots + \frac{Y_n e^{-\mu}\mu^n}{n!}$$ Signal gain (linear combination of the infinite series) (8)

$$Q_\mu = \sum_{n=0}^{\infty} Y_n \frac{e^{-\mu}\mu^n}{n!}$$ Signal gain (Summation of the infinite series) (9)

Calculating for all n

$$Q_\mu = Y_0 + 1 - e^{-\mu\eta}$$ Signal gain (closed form) (10)

and solving for η (note the signal efficiency η is derived from the measured gain Q_μ, dark count Y_0, and MPN μ; it is not dependent on the number of photons per pulse)

$$\eta = \frac{-\ln\left|1 + \mathrm{Y}_0 - Q_\mu\right|}{\mu}.$$ End-to-end efficiency (in terms of dark count and signal gain) (11)

Finally, we can estimate the photon dependent signal yields Y_n^{signal} by substituting η from Eq.End-to-end efficiency (in terms of dark count and signal gain) (11) into Eqs. Photon number dependent yield (3) and Photon number dependent efficiency (5):

$$Y_n \cong Y_0 + 1 - \left(1 - \left[\frac{-\ln\left|1 + Y_0 - Q_\mu\right|}{\mu}\right]\right)^n.$$ Photon number dependent signal yield (12)

Likewise, the end-to-end efficiency η can be calculated from the decoy state gain Q_ν and used to estimate the decoy state yields Y_n^{decoy}. The decoy and signal state photon statistics can then be compared to assess whether unauthorized photon specific interference exists on the quantum channel [28, 14].

Estimating Photon Number Dependent QBERs

The photon number dependent QBER e_n is defined as:

$$e_n = \frac{Y_0 e_0 + \eta_n e_{detection}}{Y_n}$$ Photon number dependent QBER (13)

where Y_0 is the dark count described in Eq. Dark count rate (4), e_0 is the likelihood of an error during a vacuum state (i.e., ½ since the erroneous detection results in either a 0 or 1), η_n is the photon number dependent efficiency, and $e_{detection}$ is the probability a photon was erroneously detected (i.e., due to device imperfections or misalignment) [21]. The n-photon QBER is estimated from the signal and decoy state QBERs:

$$E_\mu = \frac{\text{Number of signal bit errors}}{\text{Number of bits in error in reconciled key}}$$ Estimated n-photon QBER (using signal errors) (14)

$$E_\nu = \frac{\text{Number of decoy bit errors}}{\text{Number of bits in error in reconciled decoy buffer}}$$ Estimated n-photon QBER (using decoy errors) (15)

where the relationship between the signal state QBER E_μ and the n-photon QBER e_n is described as:

$$E_\mu Q_\mu = \sum_{n=0}^{\infty} e_n Y_n \left(\frac{e^{-\mu}\mu^n}{n!}\right).$$ Overall QBER (Summation) (16)

Calculating for all n

$$E_\mu Q_\mu = e_0 Y_0 + e_{detection}(1 - e^{-\mu\eta})$$ Overall QBER (Closed form) (17)

and solving for η

$$\eta = \frac{-\ln\left|\frac{1 - E_{\mu}Q_{\mu} - e_0\mathrm{Y}_0}{e_{detection}}\right|}{\mu}.$$

End-to-end efficiency (in terms of overall QBER and vacuum state errors) (18)

Substituting η from Eq. End-to-end efficiency (in terms of overall QBER and vacuum state errors) (18) into Eqs. Overall QBER (Closed form) (17) and Photon number dependent efficiency (5)

$$e_n = \frac{Y_0 e_0 + \left(1 - \left(1 - \left[\frac{-\ln\left|\frac{1 - E_{\mu}Q_{\mu} - e_0 Y_0}{e_{detection}}\right|}{\mu}\right]\right)^n\right) e_{detection}}{Y_n}.$$

Photon number dependent QBER (in terms of overall QBER, errors, dark count rate and yield) (19)

Likewise, the end-to-end efficiency η can be calculated from the decoy state gain Q_{ν} and used to estimate the decoy state QBERs e_n^{decoy}. The photon number dependent QBERs can then be compared to assess whether unauthorized photon specific interference exists on the quantum channel [28, 14].

qkdX Modeling Framework

An interdisciplinary team of researchers at the Air Force Institute of Technology (AFIT), the Laboratory for Telecommunication Sciences (LTS), and the US Naval Research Laboratory (NRL) developed a customizable QKD eXperimentation (qkdX) modeling and simulation framework based upon optical components modeled from real world commercial-off-the-shelf (COTS) implementations to enable a detailed performance analysis of QKD systems [17]. In particular, the team successfully constructed a notional polarization-based prepare and measure QKD architecture using the OMNeT++ simulation environment and used it to explore the effects of implementation non-idealities on QKD system performance [95]. However, the team still desired to perform verification and validation of the model, examine the model's customizability, and extend the ability to conduct detailed performance analyses of QKD systems [81]. These tasks motivated the research conducted in this thesis focused on understanding evaluating QKD systems suitability for use in communications related to national security [17].

III. Methodology

Overview

This chapter outlines the methodology used to conduct the research presented in this thesis. It provides insight into how the research questions identified in Chapter I will be answered. The research objectives will be accomplished by conducting a literature review, an independent study, model development, model validation and verification, and experimentation as explained in the sections that follow.

Literature Review

The literature review presented in Chapter II provided salient background information related to systems engineering, model based systems engineering, quantum key distribution, photon number splitting attacks, and the decoy state protocol. The material presented in the literature review answers several of the stated research questions and is required to understand and guide the research presented in this thesis.

Independent Study

An initial study will be conducted to explore both the theory and application of QKD systems using the decoy states protocol. The purpose of the study is to develop a detailed understanding of the operational aspects of the decoy state protocol so that a preliminary experiment using the qkdX notional architecture can be designed and executed using selected MBSE PMTs. The results of this study will be captured in a journal article submission to the *Journal of Defense Modeling and Simulation*, which is included as Chapter IV of this thesis. Elements of experimental design not included in this article will be included in Appendix A. Beam Splitter Experiment (BSE). Several MBSE artifacts will be created and define the “As-Is” state of the QKD system notional architecture discussed in Chapter II. The “As-Is”

DoDAF artifacts will be described in Chapter VI and use cases will be identified in Appendix C. Decoy State Enabled QKD System Use Cases.

Model Development

While the existing qkdX notional architecture was developed with the intent of being capable of implementing the Decoy State protocol, this capability was not fully developed [95]. During this research effort, MBSE will be used to define the requirements for the decoy state enabled QKD model development. This portion of the research shall begin by revising the "As-Is" MBSE artifacts into products that define the "To-Be" architecture. Lessons learned from the preliminary experiment will shape this effort and to produce updated use cases from which requirements can be derived.

Once the relevant behaviors are identified and captured using MBSE PMTs, implementation will begin. A combination of the incremental development process, the SE Vee model, and Banks *et al.'s* simulation process will be employed. Figure 14 is a depiction of this combined process as a flow chart. Requirements and use cases will be captured, elaborated, and refined through steps 1-4. Once the use cases are reviewed by Subject Matter Experts (SMEs), the model translation, i.e., implementation, will begin. Once the model is completed and verified, an experiment will be conducted to perform validation. The experimental design is fully described in Appendix B. Decoy State Enabled QKD Experiment. The results of the experiment will be captured in Chapter VI.

A summary, outlining the development process of transitioning the "As-Is" architecture into the "To-Be" model, will be submitted as an article to INCOSE's *Systems Engineering* journal and included as Chapter V. Details of the process and results which are not included in the article will be described in Chapter VI.

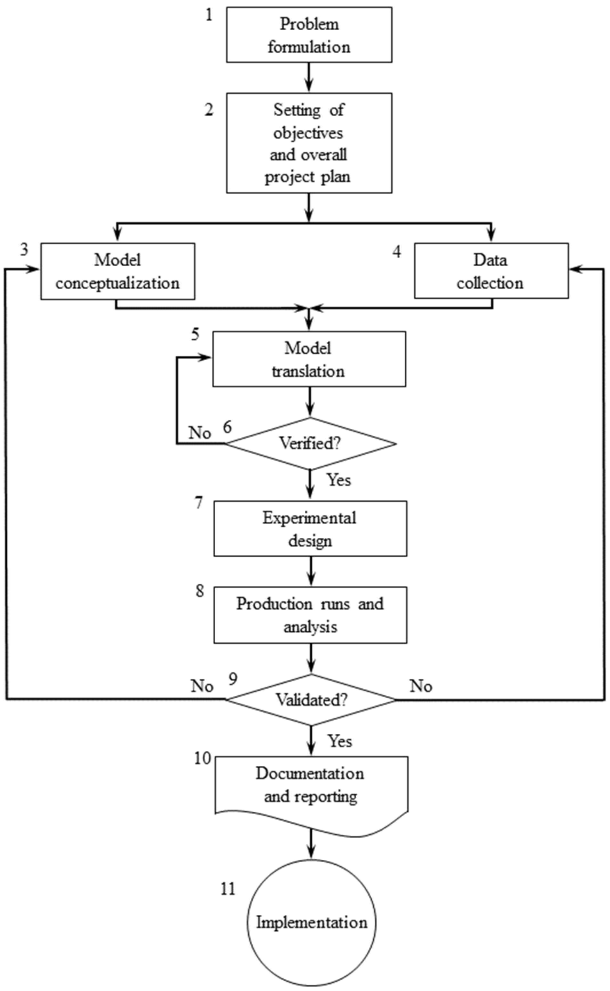


Figure 14. Tailored Simulation Model Development Process.

Model Requirements, Verification and Validation

As discussed, the SE verification process intends to confirm that system conforms to specified requirements and validation confirms that system meets the needs of the stakeholders [36]. For this research study, the primary method of verification and validation of requirements and stakeholder needs will involve observation or analysis of the simulation’s output and output data. Thus, it will be necessary to capture (and refine) measureable requirements during the course of the study.

The majority of existing notional qkdX architecture components will undergo verification in parallel with the research that will be conducted in this paper. The results of the component verification will be used to gain confidence in the results of the simulation's output. However since this effort may not conclude in time, analytical analysis will be performed as necessary to verify the output of the "To-Be" decoy state enabled QKD simulation. For example, signal and decoy gain can be calculated using the equations described in Chapter II. Expected results can be estimated prior to performing simulation runs and therefore shape measureable requirements that can be evaluated against simulation results. Initial and refined requirements will be captured respectively in Appendix E. Initial Requirements and Appendix F: Requirements. This analytical method will be combined with the results of the component verification to perform simulation model verification.

As discussed in Chapter II, validation confirms whether the model correctly represents the real system. Thus, if the decoy state enabled QKD system model is configured similarly to a realized system, the outputs of the model and system should be similar. This idea will shape the experimental design and link the experimental results to the MBSE process of validation. Elements of the requirements, verification, and validation processes will be discussed in the article included as Chapter V.

Experiment Design

The results of this experiment will be used for three purposes: 1) to establish baseline performance of the system in the absence of a photon number splitting (PNS) attack, 2) validate the model's security performance against empirical data and, 3) verify the model's ability to detect an eavesdropper performing a PNS attack. The baseline performance, i.e. yield tolerances, will be established by finding the mean and variance of single- and multi-photon signal and decoy yields over the course of a statistically significant number of simulation runs. The model's security performance, i.e., signal gain, will be compared the results reported by Chen *et al.* [94]. Once the baseline performance is established, the PNS attack will be performed and the model will attempt to detect the attack using the established tolerances. The details of the experimental design and the results will be respectively captured in Appendix B. Decoy State Enabled QKD Experiment and Chapter VI.

IV. Journal Article I: Modeling Decoy State Quantum Key Distribution Systems

This article summarizes the history and theory of employing decoy state enabled Quantum Key Distribution (QKD) systems in support of unconditionally secure communication. It discusses how non-idealities in practical systems create a vulnerability which can be exploited by an all-powerful adversary via the theoretical Photon Number Splitting attack and how the decoy state protocol attempts to mitigate this vulnerability. The main contribution of the article is it describes how a notional decoy state QKD system architecture was configured in a hybrid continuous time-discrete event simulation framework. It describes how verification and validation (V&V) were performed by comparing simulation results from the preliminary experiment to previous research which involved practical systems.

The paper was submitted to the *Journal of Defense Modeling and Simulation* (JDMS) on December 12, 2014 and is currently under review.

Modeling Decoy State Quantum Key Distribution Systems

Abstract*—Quantum Key Distribution (QKD) is an innovative technology which exploits the laws of quantum physics to generate and distribute shared secret key for use in cryptographic devices. QKD offers the advantage of "unconditionally secure" key generation with the unique ability to detect eavesdropping on the key distribution channel and shows promise for high-security applications such as those found in banking, government, and military environments. However, QKD is a nascent technology where realized systems suffer from implementation non-idealities, which may significantly impact system performance and security. In this article, we facilitate the study of the impact of these practical limitations through the modeling of a decoy state enabled QKD system. Specifically, we present a thorough background on the decoy state protocol, detailed discussion of the modeled decoy state enabled QKD system, and evidence for component and sub-system verification, as well as, multiple examples of system-level validation. Additionally, we bring attention to practical considerations associated with implementing the decoy state protocol security condition gained from these research activities.

Index Terms*—Quantum Key Distribution; Decoy State Protocol; Model and Simulation; System Security; System Performance

Authors—L .O. Mailloux, R. D. Engle, M.R. Grimaila, D.D. Hodson, J. M. Colombi, and C.V. McLaughlin

I. Introduction

Quantum Key Distribution (QKD) is the most mature application of the quantum information field and is heralded as a revolutionary technology offering the means for two physically separated parties to generate "unconditionally secure"[1] cryptographic key. Employing the laws of quantum physics, QKD systems can detect eavesdropping during the key generation process where unauthorized observation of quantum communication necessarily induces measurable errors. This work builds upon previous work [1, 2]. For detailed treatments and comprehensive reviews of QKD please see [3, 4, 5, 6].

QKD technology is emerging as an important development in the military and national defense cybersecurity solution space with research occurring in the United States Navy [7], Army [8], and Air Force [9]. Increasingly practical systems have been realized in numerous experiments and have recently become available from commercial vendors such as ID Quantique (Switzerland) [10], SeQureNet (France) [11], Quintessence Labs (Australia) [12], MagiQ Technologies (USA) [13], and Quantum Communication Technology Co., Ltd. (China) [14]. However, QKD is a nascent technology where real-world systems are constructed from non-ideal components, which may significantly impact system performance and security. As with any new technological development, and especially security technologies, one must carefully evaluate its use holistically. In the case of QKD systems, its overall security posture relies not only upon its theoretical security proofs but upon how well they are realized (i.e., designed, implemented, and operated) which few have addressed.

In this article, we provide a review of the decoy state protocol and describe our decoy state enabled QKD system model built to study practical system implementation issues. Furthermore, we provide results supporting component and sub-system verification, as well as, system-level validation against eight like experimental systems. Through these exercises we gain confidence in the subject model, as well as, additional insight into design, implementation, and operational considerations for the decoy state protocol security condition.

A comprehensive discussion of the decoy state protocol is provided in Section II, which includes a discussion of multi-photon vulnerabilities associated with practical QKD realizations. The modeled decoy state enabled QKD system is described in Section III, with emphasis on the end-to-end quantum communications. A discussion of component and sub-system verification, along with system-level validation activities and their results are provided in Section IV. Practical implementation considerations for the decoy state security condition are introduced in Section V. Conclusions and future works are offered in Section VI.

Paper submitted December 12, 2014. This work was supported by the Laboratory for Telecommunication Sciences [grant number 5743400-304-6448].

[1] Unconditionally secure in this context implies an all-power adversary can only gain a negligible amount of information on the distributed secret key without necessarily introducing detectable errors.

II. QKD Systems and the Decoy State Protocol

A. QKD Background

The genesis of QKD technology can be traced back to Stephen Wiesner, who first developed the idea of fraud-proof quantum money in the late 1960s [15]. In 1984, Charles Bennett and Gilles Brassard operationalized this concept when they proposed the first QKD protocol (i.e., BB84) where single photons, representing quantum bits (qubits), are used to securely generate shared cryptographic key [16]. Figure 1 illustrates a QKD system configured to generate shared secret key *K* for use in external encryptors to protect sensitive data, voice, or video communications. The architecture consists of a sender "Alice", a receiver "Bob", a quantum channel (i.e., an otherwise unused "dark" optical fiber), and a classical channel (i.e., a conventional networked connection). Alice and Bob each consist of a central processing unit (CPU) configured to control QKD processes, a network interface configured to control communications on the classical channel, and a quantum module configured to control communications on the quantum channel. QKD systems can be paired with traditional symmetric encryption algorithms (e.g., DES, 3DES, or AES) where the QKD-generated key is used to increase the security posture of encrypted communications through frequent re-keying. Alternatively, the QKD-generated key can be used in conjunction with the One-Time-Pad (OTP) encryption algorithm to provide unbreakable communications [17, 18].

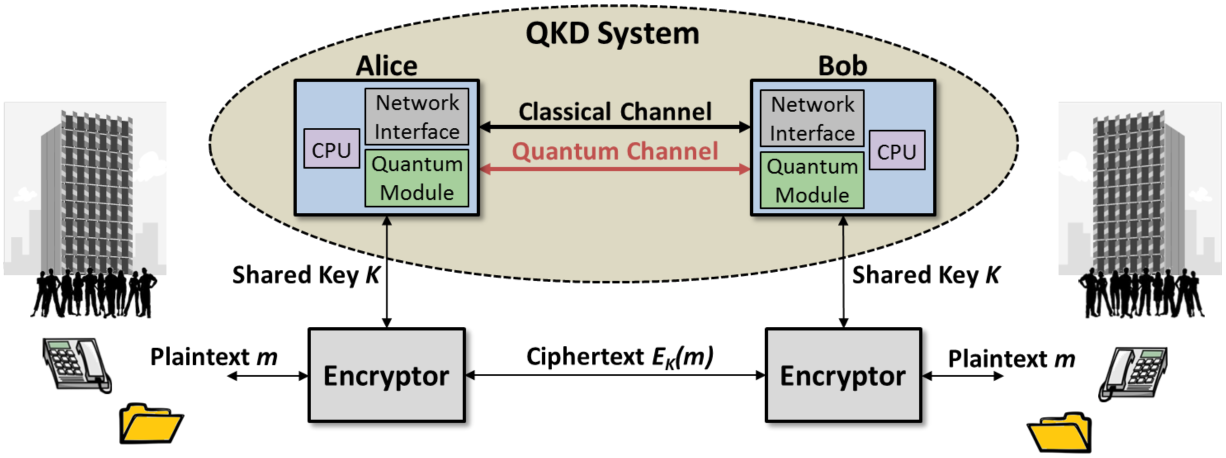


Figure 1. QKD System Context Diagram.

B. The First QKD Protocol: BB84

BB84 is a polarization-based prepare and measure QKD protocol, where four polarization states ↔, ↕, ⤡, and ⤢ are used to encode qubits as depicted in Table 1 [16, 1]. Alice prepares qubits according to a randomly selected bit value (i.e., "0" or "1") and a randomly selected basis (i.e., rectilinear "⊕" for the orthogonal polarization pair ↔, ↕ or diagonal "⊗" for the orthogonal polarization pair ⤡, ⤢). For example, when the bit value "0" and the basis "⊕" is selected, the qubit is prepared in the horizontal orientation state ↔. Likewise, when the bit value "1" and the basis "⊗" is selected, the qubit is prepared in the diagonal state ⤡. In this manner, Alice randomly prepares qubits through polarization modulation and sends them to Bob where he randomly selects a basis (⊕ or ⊗) to measure each qubit. When Alice's prepared and Bob's measured bases agree, the qubit is correctly read with a high probability. Otherwise a random result occurs as depicted in the last column of Table 1. The preparation, transmission, and measurement of qubits between Alice and Bob results is known as "quantum exchange" and inherently quantum in nature, while the remainder of the BB84 protocol (sifting, error reconciliation, and privacy amplification) is entirely classical.

Table 1. Polarization-Based Prepare and Measure Example.

The Sender "Alice" Prepares			The Receiver "Bob" Measures	
Random Bit Value	**Random Basis**	**Prepared Polarization State**	**Random Basis**	**Measured Bit Value**
0	⊕	↔	⊕	0
0	⊕	↔	⊗	0 or 1
1	⊕	↕	⊕	1
1	⊕	↕	⊗	0 or 1
0	⊗	⤡	⊕	0 or 1
0	⊗	⤡	⊗	0
1	⊗	⤢	⊕	0 or 1
1	⊗	⤢	⊗	1

During sifting, error reconciliation, and privacy amplification Alice and Bob use traditional information theory practices and communicate over the classical channel. For sifting, Bob announces the measurement bases he used to Alice, and she responds indicating where the prepared and measured bases agreed (note only the bases are revealed and not the encoded bit values). Due to the random nature of Bob's measurement basis selection, approximately 50% of the detected qubits are mismatched (i.e., erroneous) and sifted out (i.e., removed) of the secret key. Error reconciliation algorithms perform bi-directional error correction of the shared sifted keys to correct for quantum communication errors, while privacy amplification compensates for information leaked on the classical channel during error correction by producing a smaller, more secure key. Details of these processes can be found in [3, 4].

C. Multi-Photon Vulnerabilities in Realized QKD Systems

The unique nature of QKD technology makes it well-suited for high-security applications such as military, banking, and government environments, where information protection is of primary concern. The security of QKD is based on "quantum uncertainty" inherent in the BB84 protocol, where interference on the quantum channel necessarily increases the Quantum Bit Error Rate (QBER) [3, 4]. If an eavesdropper, commonly referred to as "Eve," attempts to listen on the quantum channel, she introduces additional errors and increases the QBER as captured in formal security proofs [19, 20]. Since the source of quantum bit errors cannot be known, all errors are attributed to Eve; thus the QBER must be closely monitored. Due to device imperfections and transmission errors, operational QBERs are typically 3-5% with practical security thresholds as low as 8% [4]. If the QBER threshold is exceeded, the key generation process is generally aborted, as it is assumed an adversary is interfering with the quantum key exchange [4].

However, practical design and implementation limitations can quickly undermine pristine theoretical security proofs [21]. For example, BB84 security proofs assume (and attempt to compensate for) perfect on-demand single photon sources in Alice, yet practical on-demand single photon sources are not currently available nor are they expected in the near-term [21]. Therefore, most prepare and measure architectures attenuate a classical laser pulse down from millions of photons to a Mean Photon Number (MPN) of 0.1 according to QKD security practices [3]. These sub-quantum pulses are often referred to as "weak coherent pulses" and represented probabilistically using a Poisson distribution, $P(n|\lambda) = \lambda^n e^{-\lambda}/n!$, where $\lambda =$ MPN and n represents the number of photons in an optical pulse [3]. For example, when $\lambda =$ MPN $= 0.1$, 90.48% of the pulses will have no photons, 9.05% of the pulses will have one photon, and 0.47% of the pulses will have two or more photons. This means only 1 in 10 pulses leaving Alice will actually contain a prepared qubit; the majority of pulses generated by Alice are attenuated to an energy level equivalent to zero photons.

A low MPN results in poor quantum throughput and thus low key generation rates. However, a low MPN is desirable to reduce the likelihood of multi-photon pulses, each of which exposes information about the distributed cryptographic key to a listening adversary. This multi-photon security vulnerability has been subject of a number of attacks against the quantum channel. More specifically, the Photon Number Splitting (PNS) attack was conceived to gain maximum information from this vulnerability without detection [22, 23, 24, 25]. The decoy state protocol was introduced to mitigate this attack and has since become widely implemented [26].

D. The Decoy State Protocol

The decoy state protocol represents a significant advancement in QKD technology, as the protocol is relatively cheap and easy to implement, while simultaneously increasing quantum throughput and security on the quantum channel. The decoy state protocol was introduced in 2003 [26] and quickly improved in an initial series of works [27, 28, 29, 30] and later [31, 32, 33, 34, 35, 36] by a single group of researchers (supplemental works of note include [37, 38, 39, 40]). Decoy state enabled QKD systems utilize the conventional signal state plus dedicated security states (decoy and vacuum) as described in Table 2. In this three state configuration, the signal state "μ" facilitates higher key distribution rates and greater operational distances through an increased MPN (i.e., an MPN of 0.6 is greater than the traditionally used 0.1), while the decoy state "ν" is used to increase the likelihood of detecting an attack using differential statistical analysis. Utilizing different MPNs for the signal and decoy states, allows the QKD system to detect photon number specific interference (described in Section V). The vacuum state is used to determine the detector error rate at Bob, where erroneous detections, known as "dark counts" are measured.

Table 2. Example Decoy State Protocol Configuration.

State	Purpose	MPN	Occurrence Percentage
Signal, μ	The signal state is used to transmit quantum pulses used for generating secret key.	0.6	70%
Decoy, ν	The decoy state is used to increase the likelihood of detecting photon number specific interference (i.e., the PNS attack) on the quantum channel through statistical differentiation with the signal state.	0.2	20%

Vacuum	The vacuum state is used to determine the detector error rate when no photons are present.	0	10%

An outline of the decoy state protocol is described in Table 3 [26, 27, 29] (note we've decomposed the decoy state protocol into multiple steps for clarity; it may be described differently, especially with respect to its implementation). The first step in the protocol is known as quantum exchange, where signal, decoy, and vacuum states are randomly transmitted across the quantum channel according to their unique MPN and occurrence percentage. Each pulse must have identical characteristics (e.g., wavelength, duration, etc.) other than the MPN, such that Eve cannot distinguish a decoy state from a signal or vacuum state.

After quantum exchange, Alice and Bob perform sifting where they announce the bases used to prepare and measure qubits along with each qubit's state (signal, decoy, or vacuum). For the signal and decoy states, non-matching bases are sifted from the raw secret key, where the correctly matched signal states contribute to a sifted secret key and the decoy states are used for security analysis. Sifting is not necessary on vacuum states, since every detection is considered a spurious dark count for the detector regardless of the basis.

Steps 3 and 4 are responsible for identifying, counting, and correcting errors in the signal and decoy state transmission. Counting of decoy state errors is accomplished by publically announcing and comparing the prepared and measured bit values, while error reconciliation techniques such as winnow, cascade, or Low Density Parity Check (LDPC) are used to perform bi-directional error correction of Alice's and Bob's sifted signal qubits without revealing specific bit values [3, 4].

In step 5, a number of decoy state protocol unique calculations are made from performance measurements: the signal gain Q_μ is the ratio of Bob's sifted signal detections to Alice's number of signals pulses sent; the decoy gain Q_ν is the ratio of Bob's decoy sifted detections to Alice's number of decoy pulses sent; the signal QBER E_μ is the ratio of Bob's post-sifted signal errors to the number of sifted signal pulses; the decoy QBER E_ν is the ratio of Bob's post-sifted decoy errors to the number of sifted decoy pulses; and the dark count yield Y_0 is the ratio of Bob's erroneous detections to Alice's total number of vacuum pulses sent (detailed treatment of these measurements is provided in Section II.E). Step 6 is responsible for executing the decoy state security condition which ensures there is no unauthorized interference on the quantum channel. The security check is conducted through differential statistical analysis, described in Section V.

Table 3. Decoy State Protocol.

Step	Name	Detailed Description
1	Quantum Exchange	Alice randomly encodes and transmits signal, decoy, and vacuum states (i.e., qubits) based on the prescribed occurrence percentages and MPNs, while Bob measures each qubit. This step is typically described as "quantum exchange."
2	Sifting	Bob announces the basis he used to measure each qubit. Alice responds by announcing which bases are correct, along with the state: signal, decoy, or vacuum for each pulse sent. Alice and Bob perform sifting on both the signal and decoy state qubits for matched bases.
3	Count Decoy State Errors	For the sifted decoy state qubits, Alice announces the encoded bit values to Bob over the classical channel. Bob performs a comparison of Alice's prepared bits and his measured values, and records the number of errors for use in step 5.
4	Signal State Error Reconciliation	For the sifted signal state qubits, Alice and Bob perform error reconciliation where quantum communication errors are counted and corrected for using bi-directional error correction algorithms over the classical channel. The number of corrected errors is recorded for use in step 5.
5	Calculate Gains, QBERs, and Dark Counts	The signal and decoy gains Q_μ and Q_ν, respectively are calculated from the measured number of sifted qubits and number of pulses sent. The signal QBER E_μ is calculated from the measured number of errors in the reconciled key and the number of sifted signal states, while the decoy state QBER E_ν is calculated from the measured number of errors in the sifted decoy state qubits. The number of detections measured by Bob when Alice sent a vacuum states is used to determine the system's dark count probability Y_0.
6	Perform Decoy State Security Check	A comparison of signal and decoy estimated photon number specific yields Y_n and QBERs e_n is performed. If they are not the same (i.e., within a predetermined tolerance), an eavesdropper is assumed to be listening on the quantum channel and the key is considered compromised.

E. Secret Key Rate

QKD systems are designed to securely distribute cryptographic key; however, due to higher MPNs (and therefore more multi-photon pulses) associated with the decoy state protocol, it is important to limit the secret key rate to pulses originating from Alice with a single photon. The secret key rate is described as [20, 27, 29]

$$R_{secure} \geq q\left(-Q_\mu f_{EC}\left(E_\mu\right)H_2\left(E_\mu\right) + Q_1[1 - H_2(e_1)]\right) * f_{PR} \qquad (1)$$

where the parameters are described in Table 4 [27, 29]. Of note, much of the initial decoy state research focused on increasing the secure key rate by providing improved bounds for the single photon gain Q_1 and the single photon QBER e_1 [27, 29, 31, 32, 33, 37, 38, 39, 40]. Specifically, the single photon gain Q_1 should be maximized while the single photon QBER e_1 is minimized. Furthermore, in practical QKD system implementations, the number of single photon pulses originating from Alice is directly correlated to the number of unsecure multi-photon pulses, which should be minimized. Thus there is a tradeoff between performance (i.e., signal gain) and security (i.e., the number of multi-photon pulses), where only single photon pulses contribute to unconditionally secure shared key.

Table 4. Secret Key Rate Parameters [27, 29].

Parameter Definition	Parameter Description or Calculation	
f_{PR} - laser pulse rate measured in Hz.	Typical laser pulse rates are 1-5 MHz. However, the pulse rate may vary greatly depending upon the QKD system design and implementation.	(2)
q – protocol efficiency fraction is calculated from system measurements.	$q = \frac{\text{Number of bits in error reconcilled key}}{\text{Total number of detected signal pulses}}$	(3)
Q_μ – gain of the signal state is calculated from system measurements. The decoy state gain Q_ν is similarly calculated.	$Q_\mu = \frac{\text{Number of sifted signal detections}}{\text{Total number of sent signal pulses}}$	(4)
E_μ – QBER of the signal state is calculated from system measurements. The decoy state QBER E_ν is similarly calculated.	$E_\mu = \frac{\text{Number of signal bits errors}}{\text{Number of bits in error reconcilled key}}$	(5)
$f_{EC}\left(E_\mu\right)$ – error reconciliation efficiency is determine from [41] based on the signal state QBER E_μ.	The error reconciliation efficiency varies dependent upon the signal state QBER. Typical values are ≤ 1.15.	(6)
$H_2\left(E_\mu\right)$ – uncertainty in the signal state QBER is calculated using Shannon's binary entropy limit [18].	$H_2\left(E_\mu\right) = -E_\mu log_2\left(E_\mu\right) - \left(1 - E_\mu\right)log_2\left(1 - E_\mu\right)$	(7)
Q_1 – estimated gain of pulses originating from Alice with one photon is calculated from system measurements and operational parameters.	$Q_1 \geq Q_1^{Lower} = \frac{\mu^2 e^{-\mu}}{\mu\nu - \nu^2}\left(Q_\nu^{Lower} e^{\nu} - Q_\mu e^{\mu}\frac{\nu^2}{\mu^2} - Y_0^{Upper}\frac{\mu^2 - \nu^2}{\mu^2}\right)$	(8)
$H_2(e_1)$ – uncertainty in the signal QBER for pulses originating from Alice with one photon is calculated using Shannon's binary entropy limit [18].	$H_2(e_1) = -e_1 log_2(e_1) - (1 - e_1)log_2(1 - e_1)$	(9)
e_1 – estimated QBER of pulses originating from Alice with one photon is calculated from system measurements and operational parameters, where $e_0 = 1/2$ is the dark count error rate.	$e_1 \leq e_1^{Upper} = \frac{E_\mu Q_\mu - e_0 Y_0^{Lower} e^{-\mu}}{Q_1^{Lower}}$	(10)
Y_0– dark count is calculated from systems measurements.	$Y_0 = \frac{\text{Number of dark count detections}}{\text{Number of vacuum pulses sent}}$	(11)
Y_0^{Lower}– estimated lower bound of dark count probability is calculated from system measurements for 10 standard deviations, where N_0 is the number of vacuum pulses sent by Alice and Q_0 is the measured vacuum state gain (see Eq. (4)).	$Y_0^{Lower} = Y_0\left(1 - \frac{10}{\sqrt{N_0 Q_0}}\right)$	(12)

Y_0^{Upper} – estimated upper bound of dark count probability is calculated from system measurements for 10 standard deviations, where N_0 is the number of vacuum pulses sent by Alice and Q_0 is the measured vacuum state gain (see Eq. (4))	$Y_0^{Upper} = Y_0\left(1 + \frac{10}{\sqrt{N_0 Q_0}}\right)$	(13)
Q_v^{Lower} – estimated lower bound of decoy state gain is calculated from system measurements for 10 standard deviations, where N_v is the number of decoy pulses sent by Alice.	$Q_v^{Lower} = Q_v\left(1 - \frac{10}{\sqrt{N_v Q_v}}\right)$	(14)

III. The Decoy State Enabled QKD System Model

In this section we describe the decoy state enabled QKD model built to study system performance and security. The model was constructed using the OMNeT++ Discrete Event Simulation (DES) environment with an extended library of optical components (i.e., a framework) [1]. This framework enables models to be built in a modular fashion, supporting multiple levels of abstraction to more efficiently answer simulation study research questions. The library consists of 17 optical components modeled in an event-driven paradigm with supporting control logic modeled in a process-based paradigm. Each component is configured with 12-15 (and up to 27) configurable parameters.

The focus of this research effort pertains to accurately modeling interactions between system-level behaviors and quantum phenomenon in an end-to-end decoy state quantum communication path. Thus the model is described in three distinct parts: (1) the sender, Alice's quantum module; (2) the quantum channel; and (3) the receiver, Bob's quantum module. Model assumptions are described below, where appropriate.

A. Alice's Quantum Module

Figure 2 provides a depiction of Alice's quantum module (introduced in Figure 1) designed to generate sub-quantum pulses as described in Section II.B. Alice's quantum module is designed to generate weak coherent optical pulses at a specified pulse rate, which have the desired MPN and polarization, in a reproducible manner. For example, when variations are not enabled, she first generates classical laser pulses with a MPN of 6,827,260 photons, randomly selects an encoding basis and bit value, prepares the desired polarization state (↔, ↕, ⤡, or ⤢), and attenuates the pulse's energy down to the specified MPN.

The module is shown with a classical laser source, a Polarization Modulator (PM), a Decoy State Generator (DSG), a Random Number Generator (RNG), and respective controllers (CTRL). This representation is a simplification of Alice's quantum module where additional control and timing synchronization components are not of interest and assumed to operate properly and not shown. The laser controller is configured to generate classical pulses by triggering the laser to fire in response to an electrical trigger pulse (e.g., 2 Mhz). The laser's pulse rate is configurable to match any system of interest or experiment with alternative configurations. The laser is designed to generate optical pulses representative of the ID300 commercial laser, including pulse shape, amplitude, duration, and wavelength [42, 10] (detailed discussion of the laser pulse are provided in Section IV.A and [43]). Additionally, variations in the laser pulse can be turned on or off when so desired. The laser generates unpolarized light pulses, where the PM prepares (or encode) each optical pulse according to the random bit and basis provided by the RNG. The RNG can be configured to provide pseudo random numbers through conventional RNG algorithms or quantum random numbers through a dedicated hardware interface [10].

The DSG reduces the classical pulse energy down to the specified signal, decoy, and vacuum MPNs according to the desired occurrence percentage as described with respect to Table 2. This behavior is representative of an electronic variable optical attenuator configured with a fixed optical attenuator to apply 60-100 dB depending on the desired MPN. For example, if the desired state is a signal pulse with an MPN of 0.6, the DSG will apply ~70 dB. For a decoy state of 0.08 MPN, the DSG will apply ~79 dB, and ~100 dB for the vacuum state.

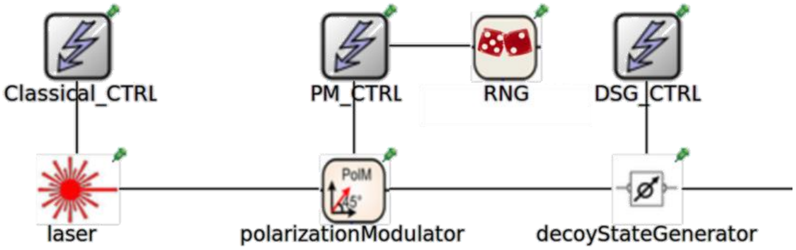


Figure 2. Modeled Alice Quantum Module.

B. *Quantum Channel*

The quantum channel is representative of Single Mode Fiber (SMF) where both the length and loss can be easily adjusted. Channel loss is specified per km where the cumulative loss is typically expressed as an efficiency (i.e., $10^{-\alpha/10}$, where $\alpha = dB\ per\ km * length$). For example, at 0.20 dB per km, the expected channel efficiency for a 50 km of SMF-28 fiber is $10^{-10/10} = 10\%$ (or conversely a 90% loss). The modeled quantum channel is also designed to account for propagation specific phenomenon such as thermal expansion, index of refraction, and chromatic dispersion. Additionally, the modeled channel is capable of simulating the effects of random physical disturbances such as bends in the fiber, vibration, or other instabilities [1]. Regarding propagation through the fiber, we assume Alice and Bob have correct basis alignment and timing control for reference frame correction and precise photon detection at Bob (i.e., polarization rotation in the quantum channel is ideally corrected and synchronized).

C. *Bob's Quantum Module*

Figure 3 is a depiction of Bob's quantum module (introduced in Figure 1) shown with a beamsplitter (BS), a half wave plate, two polarizing beamsplitters (PBS1 and PBS2), four Single Photon Detectors (SPDs), and one controller. Pulses entering Bob are randomly split by the BS (i.e., a 50/50 BS) and transmitted to PBS2 and the horizontal/vertical SPDs or reflect to the have wave plate and through PBS1 to the diagonal/antidiagonal SPDs. This configuration is commonly referred to as a "passive basis selection," where Bob's random basis selection intrinsically occurs at the 50/50 BS.

The PBSs are modeled similarly to the BS; however, they are designed to split pulses into orthogonal polarization states (i.e., the horizontal or vertical state). For example, when a pulse is prepared in the rectilinear basis and sent to the PBS, it is split according to its encoded polarization state, where horizontally encoded qubits are transmitted and vertically encoded qubits are reflected. Alternatively, when a pulse is prepared in the diagonal basis and sent to the PBS, it will be randomly split towards either output as determined by the adjacent detector and illustrated by Bob's measured bit value in Table 1.

With respect to pulses transmitted by the BS to PBS2, they are split into the horizontal state ↔ or vertical state ↕ and sent to their respective SPDs. The reflected pulses are directed towards a half wave plate that induces a $\pi/4$ polarization rotation to correctly align diagonally prepared pulses into the rectilinear basis and then to PBS1 which further distinguishes the antidiagonal ⤡ and diagonal ⤢ polarization states for detection at their respective SPDs. In this manner, rectilinearly prepared qubits which randomly proceed through PBS2 will be correctly detected at the vertical and horizontal SPDs with a high degree of accuracy, diagonally prepared qubits which randomly proceed through PBS1 will be correctly detected at the antidiagonal and diagonal SPDs with a high degree of accuracy, and pulses which are sent to the mismatch measurement basis will result in a random bit value that will be sifted out.

In the modeled architecture, the SPDs are responsible for probabilistically determining the detection of weak coherent pulses; thus simulating the uncertainty inherent in quantum phenomenon [44, 45]. Specifically, for each pulse arriving at an SPD a probabilistic determination is made for how many photons arrived according to its MPN and Poisson distribution [3]. Note: after propagating through the quantum channel and Bob's architecture the pulse's MPN is significantly lower than when it left Alice (e.g., a signal state MPN of 0.65 propagating though 50 km of fiber with 0.20 dB loss per km, and a passive basis selection architecture with 3 dB loss will have an MPN of ~0.0325). The probabilistic design of the SPDs enables quantum phenomenon such as interference between photons and the splitting of a single photon to be accurately simulated in the modeled architecture.

The detection CTRL is configured to precisely "gate" the SPDs to reduce noise (i.e., dark counts and after pulsing) while detecting single photon transmissions. This means the SPDs change from a classical detection state to a heightened "Geiger" state configured to detect the arrival of single photons. Typical gating periods are on the order of nanoseconds (i.e., 10^{-9}), which are controlled by the detection controller and assumed to be ideal in this model. Bob's SPDs take into account detector efficiency (e.g., 10%), dark count probability (e.g., 1.0E-6), after pulsing probability (e.g., 1.0E-6), photon arrival times, and detector recovery times. Modeling these parameters provides a baseline for modeling realized decoy state QKD systems.

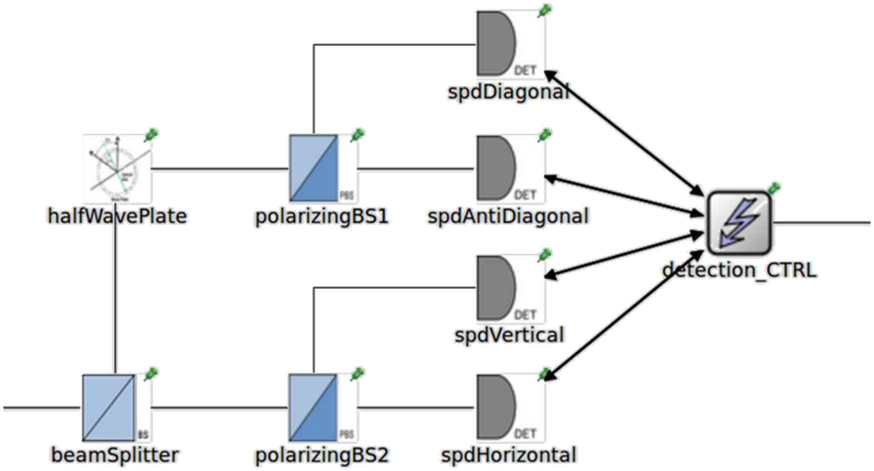


Figure 3. Modeled Bob Quantum Module.

D. *Decoy State Protocol Logic*

The decoy state protocol as outlined in Table 3 is implemented through the described physical model and control logic within Alice and Bob. In our model, Alice is generally implemented as the protocol master and Bob as the slave; however, these roles can change depending on how the desired functionality is modeled. In some cases it may be more advantageous to have Bob, as the receiver, control processes or sub-processes to reduce the number of classical communications.

To perform quantum exchange (step 1), Alice and Bob use a flexible data frame structure defined as a timing reference pulse followed by a configurable number of weak coherent pulses (e.g., 100 or 1000) with adjustable spacing between both frames and pulses. Alice randomly distributes signal, decoy, and vacuum pulses within each frame according to their occurrence percentages and records the frame number, bit position, prepared basis, encoded bit value, and state (signal, decoy, or vacuum) for each pulse generated. Likewise, upon detection Bob stores the frame number, bit position, measurement basis, and detected value. Quantum exchange ends when Bob achieves the specified number of detections (signal, decoy, and vacuum). The remainder of protocol control occurs over the classical channel using abstract message traffic passed between Alice and Bob including large memory arrays to quickly and easily facilitate information exchange for the remaining classical steps.

During sifting (step 2), Bob publically announces the bases he used to detect each qubit to Alice, and she responds by announcing which bases are correct and identifies the signal, decoy and vacuum states. Alice and Bob perform sifting on both the signal and decoy state detections to identify matched prepare and measure bases. Additionally, for each decoy state qubit, Alice announces the encoded bit value to Bob. The announced decoy state bases and bit values are used to calculate the number of errors in the decoy state transmission (Step 3). Furthermore, Alice calculates the dark count Y_0 using the total number of detections during vacuum states as defined in Eq. (11).

Next, Alice and Bob perform error reconciliation (step 4) of the sifted key to correct for quantum communication errors. They first sacrifice an adjustable percentage of the signal state bits (e.g., 25%) to estimate the error rate. Specifically, Alice randomly chooses a specified percentage (e.g., 25%) of the post-sifted signal bits and sends them (frame number, bit number, and value) to Bob. He calculates the estimated error rate and sends it back to Alice. If the estimated error rate is higher than a user defined threshold (e.g., 25%), the protocol is aborted. If the estimated error rate is below the threshold, the QKD system formally begins error reconciliation.

The simulation framework provides the ability to use conventional error reconciliation algorithms such as winnow, cascade, and LDPC where the algorithm's efficiency is based on the estimated error rate. For example, the error rate can be used to determine block correction sizes and influences the number of communications necessary for error reconciliation. These algorithms are generally considered resource intensive, so we implemented a "perfect" error reconciliation algorithm, which quickly counts and corrects errors without unwanted complexity. Since we are not studying error reconciliation, we anticipate using the perfect error reconciliation most often because of its speed advantage while running large simulation trails. During perfect error reconciliation, Alice sends her key to Bob who determines the number of errors in his key and returns the results back to Alice.

In step 5, the decoy state protocol specific measurements (i.e., the gains Q_μ and Q_ν and QBERs E_μ and E_ν as defined in Eqs. (4) and (5)) are calculated by Alice. These values are exported from the simulation framework for use in analytical tools. Using third party tools allows us to quickly analyze numerous simulation trials and facilitates system-level testing. Furthermore, analytical tools are required for statistical analysis of the decoy state security condition.

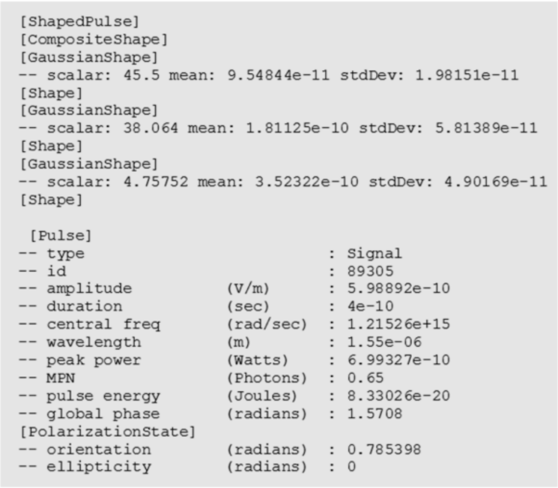
In step 6, the signal and decoy photon number dependent yields Y_n^{signal} and Y_n^{decoy} along with the photon number dependent QBERs $e_n^{signal}$and e_n^{decoy} are calculated. These values are estimated from the measured gains Q_μ and Q_ν, QBERs E_μ and E_ν, and dark count Y_0, and compared to determine if an eavesdropper is listening on the quantum channel. Details of this step are provided in Section V.

IV. Verification and Validation of the Decoy State Enabled QKD System Model

In this section we describe our Verification & Validation (V&V) approach and provide results of our component, sub-system, and system-level test activities. Our V&V work is primarily based on the approaches and specific activities discussed in [46, 47, 48]. In general, we confirm the model was built correctly against expected analytical results, and validate performance against multiple experimental systems. More specifically, component verification is conducted against commercial specifications, where modeled behaviors and operational parameters are evaluated for each component by assessing its transformation of optical pulses. Sub-system verification is conducted using analytical means to test the integration of multiple components by assessing their combined transformations against expected pulse MPNs and polarization orientation. Since, the modeled QKD system and its verification is primarily concerned with the transformation of optical pulses, we provide an abbreviated description of the optical pulse model for the reader before we proceed with our detailed discussion of verification activities. Details of the optical pulse model can be found in [43].

A. Optical Pulse Model

Figure 4 displays a textural representation of the modeled pulse output generated by Alice. She creates pulses according to the state (i.e., signal, decoy, or vacuum), prescribed MPN, occurrence percentage, and polarization encoding. The pulse model is abstracted using a classical electromagnetic wave representation, $\vec{E}$, with a time-dependent power envelope, $\sqrt{G(t)}$, to model each pulse $\vec{E}(t) = \sqrt{G(t)} E_o e^{i\omega_0 t} e^{i\theta} \begin{bmatrix} cos\alpha \\ (sin\alpha)e^{i\phi} \end{bmatrix}$ [43]. For economy of simulation, the power envelope $\sqrt{G(t)}$ is represented as a composite shape using the sum of three Gaussian functions each with a scalar, mean, and standard deviation. The pulse model captures the electric field amplitude E_0, a specified duration used to identify the pulse interval time Δt, the central frequency ω_o, and the global phase θ. Method calls calculate pulse wavelength, peak power, MPN, and pulse energy from the pulse's shape, amplitude, duration, and central frequency. The pulse's encoded polarization state is captured by the orientation angle α with respect to the x axis, while ellipticity is used to represent the relative phase delay ϕ for phase-based QKD encoding. This wave representation allows for efficient calculation of optical effects such as propagation, dispersion, attenuation, or interference by individual optical components [43]. Each pulse also has a unique identifier to track its simulation and verify its expected behaviors.



```
[ShapedPulse]
[CompositeShape]
[GaussianShape]
-- scalar: 45.5 mean: 9.54844e-11 stdDev: 1.98151e-11
[Shape]
[GaussianShape]
-- scalar: 38.064 mean: 1.81125e-10 stdDev: 5.81389e-11
[Shape]
[GaussianShape]
-- scalar: 4.75752 mean: 3.52322e-10 stdDev: 4.90169e-11
[Shape]

 [Pulse]
-- type                           : Signal
-- id                             : 89305
-- amplitude          (V/m)       : 5.98892e-10
-- duration           (sec)       : 4e-10
-- central freq       (rad/sec)   : 1.21526e+15
-- wavelength         (m)         : 1.55e-06
-- peak power         (Watts)     : 6.99327e-10
-- MPN                (Photons)   : 0.65
-- pulse energy       (Joules)    : 8.33026e-20
-- global phase       (radians)   : 1.5708
[PolarizationState]
-- orientation        (radians)   : 0.785398
-- ellipticity        (radians)   : 0
```

Figure 4. Modeled Optical Pulse.

B. Component Verification

Component verification was conducted for each of the modeled optical components (laser, polarization modulator, electronic variable optical attenuator, fixed optical attenuator, fiber channel, beamsplitter, polarizing beamsplitter, half-wave plate, and various other components not shown such as a bandpass filter, circulator, in-line polarizer, isolator, optical switch 1x2, polarization controller, and wave division multiplexer). We used analytical means to verify the modeled results according to industry specifications and subject matter experts. Specifically, the modeled components were implemented in C++ and verified using Python according to the testing methodology detailed in Figure 5. The testing approach is intended to provide a flexible capability for verification of optical components, where new or modified components can be easily added and tested.

The testing framework is focused on verifying the primary behaviors of each component by ensuring the expected transformations occur correctly. This is accomplished by creating a linked list of optical pulses (i.e., the test input) according to the pulse parameters of interest (i.e., shape, amplitude, duration, central frequency, global phase, orientation, and ellipticity). The input list of optical pulses is sent to both the modeled component and the truth data calculation. Each component transforms the optical pulses according to specified behaviors and configuration parameters, resulting in an output list of optical pulses. Likewise, the truth data calculation transforms the list of optical pulses according to commercial specifications. A comparison of the model and expected truth data is performed for each transformed pulse in the list, where each device will either pass the suite of tests if the pulses match or fail if the pulses do not match. The testing framework accounts for each modeled optical component in a systematic way, which allows users to quickly and repeatedly test component behaviors over their operational ranges.

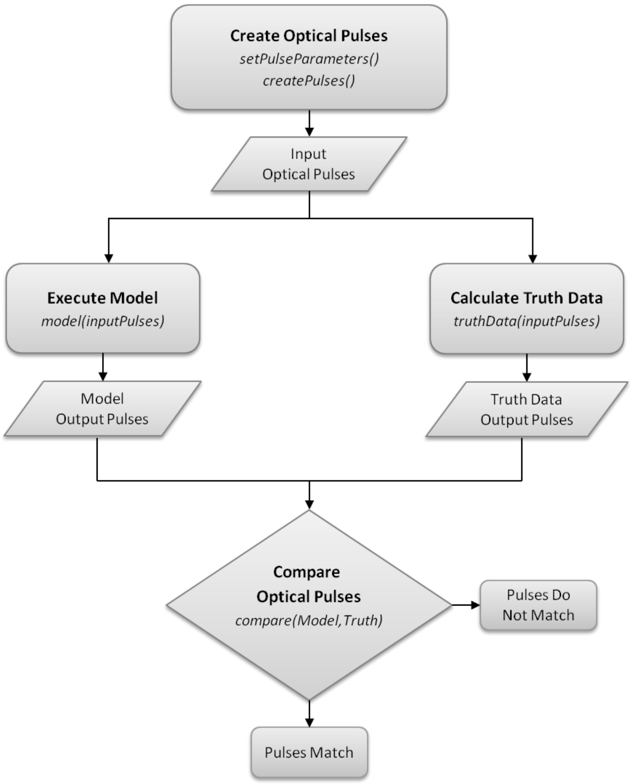


Figure 5. Component Testing Methodology.

An example of this testing methodology is provided in Table 5, where five input pulses with varying orientations (0, $\pi/8$, $\pi/4$, $3\pi/8$, and $\pi/2$) and a fixed MPN of 0.6500 are shown propagating through a polarizing beamsplitter (note this example is greatly simplified for the reader, actual tests included 10,000s of points). The polarizing beamsplitter is modeled with one input and two outputs configured to split the input pulse according to the transmitted polarization (i.e., an orientation of 0) and the reflected polarization (i.e., an orientation of $\pi/2$). The device is designed to project the input pulse's energy (i.e., MPN) onto the transmitted and reflected polarization states. For example, since the first pulse's orientation is 0, the entirety of the pulse's energy will be transmitted and none will be reflected as demonstrated by both the modeled and truth data output results. Likewise, when considering the fourth pulse, we expect a small portion (~15%) of the energy to be observed on the transmitted port and a large amount (~85%) on the reflected port. This type of testing regime was automated in the framework and repeated thousands of times across pertinent operational and configuration parameters for each optical device in the component library. These components are modeled with ideal behaviors that result in perfect agreement to the expected values. Non-ideal behaviors caused by device imperfections or misuse can also be simulated. For example, each component is modeled with three operational states: normal, degraded, and damaged, where the device can enter the degraded or damaged state due to overheating or overpowering with optical light. These non-ideal states negatively impact the devices' behavior(s).

Table 5. Polarizing Beamsplitter Verification.

Input Pulses			**Modeled Output Pulses**		**Truth Data Output Pulses**	
Number	MPN	Orientation	Transmitted MPN (Orientation = 0)	Reflected MPN (Orientation $=\pi/2$)	Transmitted MPN (Orientation = 0)	Reflected MPN (Orientation $=\pi/2$)
1	0.6500	0	0.6500	0.0000	0.6500	0.0000
2	0.6500	$\pi/8$	0.5548	0.0952	0.5548	0.0952
3	0.6500	$\pi/4$	0.3250	0.3250	0.3250	0.3250
4	0.6500	$3\pi/8$	0.0952	0.5548	0.0952	0.5548
5	0.6500	$\pi/2$	0.0000	0.6500	0.0000	0.6500

C. *Sub-System Verification*

Sub-system verification was conducted using the experimental configuration as reported in Chen *et al.*, where the USTC-Xinglin QKD architecture employed a signal MPN of 0.65, a decoy MPN of 0.08, a measured channel loss of 4.5 dB, and a reported 3.5 dB loss in Bob [49]. From the sub-system perspective there are three points of interest in the communication system: (1) Alice's output; (2) Bob's input; and (3) Bob's SPDs. For each point, we used analytical means to verify the state of the optical pulse for the expected and modeled MPNs and polarization orientation. Table 6 depicts the sub-system verification results where two pulses, a signal and a decoy, are tracked throughout the simulation using a unique identifier. We also verified the occurrence percentages (75%, 12.5%, and 12.5% for the signal, decoy, and vacuum states respectively).

Table 6. Sub-System Verification of Pulse MPN and Orientation.

Alice's Output				**Bob's Input**			**Bob's SPDs**		
Pulse ID	**Type**	**MPN**	**Orientation**	**Channel Loss**	**MPN**	**Orientation**	**Bob's Loss**	**MPN**	**Orientation**
4396	Signal	0.650000	π/2	4.5 db	0.179025	π/2	3.5 dB	0.079967	π/2
Expected result		*0.650000*	*π/2*		*0.179025*	*π/2*		*0.079967*	*π/2*
34811	Decoy	0.080000	π/4		0.022034	π/4		0.009842	π/4
Expected result		*0.080000*	*π/4*		*0.022034*	*π/4*		*0.009842*	*π/4*

Each pulses' MPN was verified against analytical calculations proving the modeled components were integrated properly. Verifying these three points ensures our optical pulse is propagating correctly through the end-to-end quantum path in preparation for system-level validation. Finally, we verified the expected signal, decoy, and vacuum occurrence percentages at Alice's output with an average of 74.99978%, 12.50350%, and 12.49672% for 120 trials of 28,000 detections.

D. *System Validation*

We conducted validation against similar systems, providing valid ranges of operation for our decoy state QKD model. Specifically, we chose five systems representative of a short operational distance (i.e., 20 km) [31, 50, 49, 51, 52] and three of a longer practical distance (i.e., 50-60 km) [53, 32, 54]. The details reported in these articles, including architectural design, configuration parameters, and performance results lend themselves to a thorough validation effort. Operational parameters and results from each of the eight systems are presented in Table 7, along with results from representative models. Each of the simulated models include system implementations details such as signal and decoy MPNs, occurrence percentages, channel losses, receiver losses, detector efficiency, and detector dark count. The simulation results are reported for 60 trials of 50,000 detections each for a total of nearly 500,000,000 pulses sent per experimental treatment to provide statistically significant secret key rates R_{secure}.

Table 7. Validation Results.

QKD System	Distance (km)	MPN μ / ν	Percent μ / ν	Channel Loss (db)	Receiver Loss (dB)	Detector Efficiency (%)	μ Gain**	ν Gain**	μ QBER	ν QBER	R_{secure}
Chen *et al.* [49]	20	0.60 / 0.08	75.0 / 12.5	5.6	3.5	10	6.36E-3	8.61E-4	1.44E-2	7.80E-2	4.10E-4
Model Results							*6.14E-3*	*9.61E-4*	*1.21E-2*	*6.57E-2*	*6.68E-4*

Dixon *et al.* [50]	20	0.55 / 0.10	80 / 16	4.012	4.2*	10	8.680E-3	1.970E-3	2.530E-2	Not reported	9.909E-4*
Model Results							*6.775E-3*	*1.681E-3*	*1.44E-2*	*1.68E-2*	*11.755E-4*
Yuan *et al.* [52]	25	0.425 / 0.204	75 / 25	4.7	2.5	10*	8.500E-3	4.000E-3	1.720E-2	2.74E-2	8.806E-4*
Model Results							*6.392E-3*	*3.275E-3*	*1.44E-2*	*2.44E-2*	*12.013E-4*
Dynes *et al.* [51]	20	0.55 0.098	93.0 / 6.2	4.0*	2.5	10	1.270E-2	2.340E-3	1.80E-2	6.200E-2	1.464E-3*
Model Results							*9.296E-3*	*1.968E-3*	*1.24E-2*	*3.96E-2*	*2.093E-3*
Zhoa *et al.* [31]	15	0.8 / 0.12	90 / 10	3.15	6.4*	10*	8.757E-3	1.360E-3	9.536E-3	2.689E-2	3.588E-4
Model Results							*6.039E-3*	*1.045E-3*	*9.219E-3*	*3.890E-2*	*5.586E-4*
Dixon *et al.* [53]	50	0.5 / 0.1	98.83 / 0.78	10	3.5	16.5	4.00E-3	1.00E-3	3.85E-2	1.30E-2	2.590E-04
Model Results							*2.39E-3*	*9.76E-4*	*9.61E-2*	*2.33E-1*	*4.536E-4*
Zhoa *et al.* [32]	60	0.55 / 0.152	63.5 / 20.3	12*	3.0*	10	1.81E-3	5.47E-4	3.05E-2	7.78E-2	1.018E-4
Model Results							*1.50E-3*	*4.93E-4*	*2.51E-2*	*7.86E-2*	*1.6996E-4*
Peng *et al.* [54]	75	0.6 / 0.2	50 / 40	15.15	7.775	6.5	2.08E-4	7.53E-5	3.23E-2	9.04E-2	1.1430E-5
Model Results							*2.70E-4*	*1.04E-4*	*2.91E-2*	*7.30E-2*	*1.8560E-5*

* Estimated values from published results.

** Simulations were conducted at 100% detector efficiency to reduce simulation time, thus we scaled our reported gains according to the experimental detector efficiencies (e.g., 10%).

Each QKD system is entirely unique, where the reported results are dependent upon dozens of operational and implementation details including calibration of the optical channel, programming of device controllers, various device imperfections, and the changing operational environment. Therefore, in these validation activities, we do not attempt to precisely match the reported performance but rather we attempt to replicate their results with a reasonable level of accuracy to justify the use of our model. That is, we prove sufficient accuracy for the intended purpose of studying the decoy state protocol. Specifically, by modeling the operational configuration parameters of Table 8, as well as the reported dark count and after pulse probabilities, we achieve simulation results within the same order of magnitude as the reported experimental system gains Q_μ, Q_ν, QBERs E_μ, E_ν, and the secret key rate R_{secure}.

By modeling and simulating architectural details such as reported propagation distances, losses, MPNs, occurrence percentages, and detector efficiencies we have validated our modeled decoy state enabled QKD system for operational distances of ~20 km and ~60 km. Our verification activities ensured the model was built correctly, while our validation activities provide objective evidence that we built the correct model. These V&V activities provide analytical and experimental evidences for the accurate modeling of decoy state QKD systems to support further research and performance analysis.

V. Considerations for the Decoy State Protocol Security Condition

While the described V&V activities described herein were primarily focused on ensuring the correctness of modeled behaviors, they also provide insight into practical considerations for implementing the decoy state protocol security condition which have not been explicitly addressed in other literature. In this section, we first introduce the security condition and then discuss practical implementation issues and initial findings from modeling, verifying, and validating these behaviors. Specifically, we focus on operational parameters and design decisions affecting the decoy state protocol security condition.

A. The Decoy State Protocol Security Condition

Recall, the decoy state protocol security condition was introduced to detect photon specific interference caused by the PNS attack, where the general premise is simply that channel losses and induced errors should remain constant over a given quantum channel. Formally, the decoy state protocol security condition is described as [27, 29]

$$Y_n = Y_n^{signal} = Y_n^{decoy} \tag{15}$$

$$e_n = e_n^{signal} = e_n^{decoy} \tag{16}$$

where the photon number dependent yield Y_n is defined as the conditional probability that Bob detects an optical pulse given Alice sent an n-photon pulse and the photon number dependent QBER e_n is defined as the conditional probability an error occurred given Bob detects the n-photon pulse from Alice [27, 29].

The security condition asserts the yields Y_n and QBERs e_n for the signal and decoy states should always be the same (i.e., within prescribed tolerances) for each n-photon pulse. This is because the quantum transmission efficiency and error rate are fixed for the quantum channel. For example, when no interference is present $Y_1^{signal} = Y_1^{decoy}$ should always be true because the end-to-end transmission efficiency does not change; thus the likelihood of any pulse with one photon propagating from Alice to Bob is identical, regardless of state. Likewise, the QBERs e_n are based on device and alignment imperfections over the quantum channel and should always be the same for each photon number. Thus Y_n and e_n vary only with respect to the photon number per pulse (see details below).

B. Implementing the Decoy State Security Condition

There are quite a few nuances to implementing the security condition. First, in practical QKD implementations, the photon number dependent yields $Y_n^{signal}, Y_n^{decoy}$ and QBERs $e_n^{signal}, e_n^{decoy}$ are merely estimates of expected values. They are estimates because QKD realizations often Avalanche Photodiode Detectors (APDs), which are threshold detectors that cannot determine the specific number of photons received in each optical pulse. Therefore, Bob cannot precisely measure the n-photon yields Y_n or QBERs e_n; they are estimated from the measured state gains $Q_\mu, Q_{v,}$ and QBERs E_μ, E_v. First, we'll address the photon dependent yields Y_n and then the QBERs e_n.

C. Estimating Photon Number Dependent Yields

The photon number depended yield Y_n is defined as [27, 29]

$$Y_n = Y_0 + \eta_n - Y_0\eta_n \cong Y_0 + \eta_n \qquad (17)$$

where Y_0 is the dark count described in Eq. (11), η_n is the photon number dependent efficiency, and the joint probability $Y_0\eta_n$ is typically disregarded because it is very small (i.e., 10^{-9}). Treating each photon independently, the photon dependent efficiency η_n is based on the number of photons n in each pulse and the end-to-end efficiency η (commonly expressed as both the likelihood of successful propagation or in the negative sense as loss measured in dB)

$$\eta_n = 1 - (1 - \eta)^n \qquad (18)$$

where

$$\eta = \eta_{QuantumChannel}\eta_{Bob}\eta_{detector}. \qquad (19)$$

The overall efficiency η is the product of the quantum channel efficiency $\eta_{QuantumChannel}$, the receiver's efficiency η_{Bob}, and the efficiency of the Bob's detector $\eta_{detector}$. The quantum channel efficiency is often approximated as $\eta_{QuantumChannel} = 1 - 10^{-\alpha/10}$, where $\alpha = 0.20\ dB/km * \ell ength$ [3] or a measured loss in dB. Bob's efficiency η_{Bob} is typically measured and represented as loss in dB, while his detector efficiency $\eta_{detector}$ is device dependent and typically represented as a percentage (e.g., 10%). When studying the decoy state protocol, the protocol efficiency $\eta_{protocol}$ must also be considered

$$\eta = \eta_{QuantumChannel}\eta_{Bob}\eta_{detector}\eta_{protocol}. \qquad (20)$$

The addition of the protocol efficiency $\eta_{protocol}$ accounts for the signal state occurrence percentage (i.e., < 100%). For example, in the Chen *et al.* QKD system, only 75% of the transmitted pulses contribute to overall signal gain, while the decoy state account for 12.5%, and vacuums account for the remaining 12.5%. These operational parameters are part of an overall system trade space which directly impact system performance.

In some cases the total efficiency is known and can be used to calculate the yields $Y_n^{signal}, Y_n^{decoy}$; however, assuming the channel is untrusted, the efficiency can be calculated from the measured gains. The relationship between the signal gain Q_μ and the efficiency η is described in Eqs. (21) - (24)

$$Q_\mu = Y_0 e^{-\mu} + Y_1 e^{-\mu}\mu + \frac{Y_2 e^{-\mu}\mu^2}{2} + \cdots + \frac{Y_n e^{-\mu}\mu^n}{n!} \qquad (21)$$

$$Q_\mu = \sum_{n=0}^{\infty} Y_n \frac{e^{-\mu}\mu^n}{n!}. \tag{22}$$

Calculating for all n

$$Q_\mu = Y_0 + 1 - e^{-\mu\eta} \tag{23}$$

and solving for η (note the signal efficiency η is derived from the measured gain Q_μ, dark count Y_0, and MPN μ; it is not dependent on the number of photons per pulse)

$$\eta = \frac{-\ln\left|1 + \mathrm{Y}_0 - Q_\mu\right|}{\mu}. \tag{24}$$

Finally, we can estimate the photon dependent signal yields Y_n^{signal} by substituting η from Eq. (24) into Eqs. (17) and (18)

$$Y_n \cong Y_0 + 1 - \left(1 - \left[\frac{-\ln\left|1 + Y_0 - Q_\mu\right|}{\mu}\right]\right)^n. \tag{25}$$

Likewise, the end-to-end efficiency η can be calculated from the decoy state gain Q_ν and used to estimate the decoy state yields Y_n^{decoy}. The photon number dependent yields can then be compared to assess whether unauthorized photon specific interference exists on the quantum channel.

D. Estimating Photon Number Dependent QBERs

The photon number dependent QBER e_n is defined as [27, 29]

$$e_n = \frac{Y_0 e_0 + \eta_n e_{detection}}{Y_n} \tag{26}$$

where Y_0 is the dark count described in Eq. (11), e_0 is the likelihood of an error during a vacuum state (i.e., ½ since the erroneous detection results in either a 0 or 1), η_n is the photon number dependent efficiency, and $e_{detection}$ is the probability a photon was erroneously detected (i.e., due to device imperfections or misalignment, typically ~1% [25]). The n-photon QBER is estimated from the state QBERs from Eq. (5), where the relationship between the signal state QBER E_μ and the n-photon QBER e_n is described as [27, 29]

$$E_\mu Q_\mu = \sum_{n=0}^{\infty} e_n Y_n \left(\frac{e^{\mu}\mu^n}{n!}\right). \tag{27}$$

Calculating for all n

$$E_\mu Q_\mu = e_0 Y_0 + e_{detection}\left(1 - e^{-\mu\eta}\right) \tag{28}$$

and solving for η

$$\eta = \frac{-\ln\left|\frac{1 - E_\mu Q_\mu - e_0 \mathrm{Y}_0}{e_{detection}}\right|}{\mu}. \tag{29}$$

Substituting η from Eq. (29) into Eqs. (28) and (18)

$$e_n = \frac{Y_0 e_0 + \left(1 - \left(1 - \left[\frac{-\ln\left|\frac{1 - E_\mu Q_\mu - e_0 Y_0}{e_{detection}}\right|}{\mu}\right]\right)^n\right) e_{detection}}{Y_n}. \qquad (30)$$

Likewise, the end-to-end efficiency η can be calculated from the decoy state gain Q_ν and used to estimate the decoy state QBERs e_n^{decoy}. The photon number dependent QBERs can then be compared to assess whether unauthorized photon specific interference exists on the quantum channel.

E. Security Condition Example

Assuming an eavesdropper is not going to knowingly introduce errors, Eve's detectability is primarily dependent upon the system's ability to detect changes in the yields Y_n; therefore, we'll examine the estimated photon number dependent yields Y_n, Y_n^{signal}, and Y_n^{decoy} as shown in Table 8. For each row, the n-photon yield is calculated using the derived signal or decoy state efficiencies based on the Chen et al. USTC-Xinglin QKD system. In the first column, the yields $\mathrm{Y_n}$ are calculated from the reported channel efficiency at 5.6 dB, Bob's efficiency at 3.5 dB, detector efficiency at 10%, and protocol efficiency at 75%, where the total efficiency $\eta = 0.009541$ per Eq. (20). Next, the signal yields $Y_n^{signal}$are calculated from the measured signal gain Q_μ using Eq. (24), where $\eta = -\ln|1 + 1.0\mathrm{E}^{-4} - 6.36\mathrm{E}^{-3}|/0.65 = 0.009661$. Lastly, the decoy yields Y_n^{decoy} are calculated from the measured decoy gain where $\eta = -\ln|1 + 1.0\mathrm{E}^{-4} - 8.61\mathrm{E}^{-4}|/0.08 = 0.009516$.

Table 8. Example Photon Dependent Yields.

Photon Number	Yield Y_n $(\eta = 0.009541)$	Signal Yield Y_n^{signal} $(\eta = 0.009661)$	Decoy Yield Y_n^{decoy} $(\eta = 0.009516)$
n = 1	0.009641	0.009761	0.009616
n = 2	0.019090	0.019329	0.019042
n = 3	0.028450	0.028804	0.028378
n = 4	0.037720	0.038188	0.037625
n = 5	0.046902	0.047481	0.046784

As the reader can ascertain, the photon specific yields are similar but not the same in all cases. We now turn our attention to understanding these differences in the form of implementation considerations and allowable tolerances.

F. Security Condition Considerations

In order to effectively implement the decoy state protocol, there are a number of design, implementation, and operational considerations to consider, including signal and decoy MPNs, occurrence percentages, propagation distances, total losses, desired detection sensitivity, the number of signal, decoy, and vacuum detections, and their resultant confidence intervals, amongst others. Therefore, when attempting to execute the decoy state protocol security condition, there needs to be additional consideration for implementation non-idealities and operational tolerances such that the security condition becomes

$$Y_n = Y_n^{signal} \pm \Delta = Y_n^{decoy} \pm \Delta \qquad (31)$$

$$e_n = e_n^{signal} \pm \Delta = e_n^{decoy} \pm \Delta \qquad (32)$$

where Δ describes the tolerances associated with the detection of Eve. For example, in 120 trials designed to collect qubits for error reconciliation blocks of 10,000 bits (i.e., 28,000 detections when accounting for 75% signal state and 50% sifting), we observed variations in both the signal and decoy photon dependent yields as depicted in Figure 6 and reported in Table 9. This is primarily because the photon number dependent yields for each state are based on the number of sifted bits (i.e., 50% of the number of detections) and total number of pulses sent in each trial. For both cases, the signal state has tighter bounds than the decoy state. These variations, as well as variations in dark count and MPNs, directly impact the estimated yields. While the impact of fluctuations in laser sources (i.e., the MPNs) has been addressed for the secret key rate [33, 34, 35], these fluctuations have not been addressed for the decoy state security condition. Accuracy of the signal and decoy MPNs is critical for performing the security check as identified in Eqs. (24) and (25).

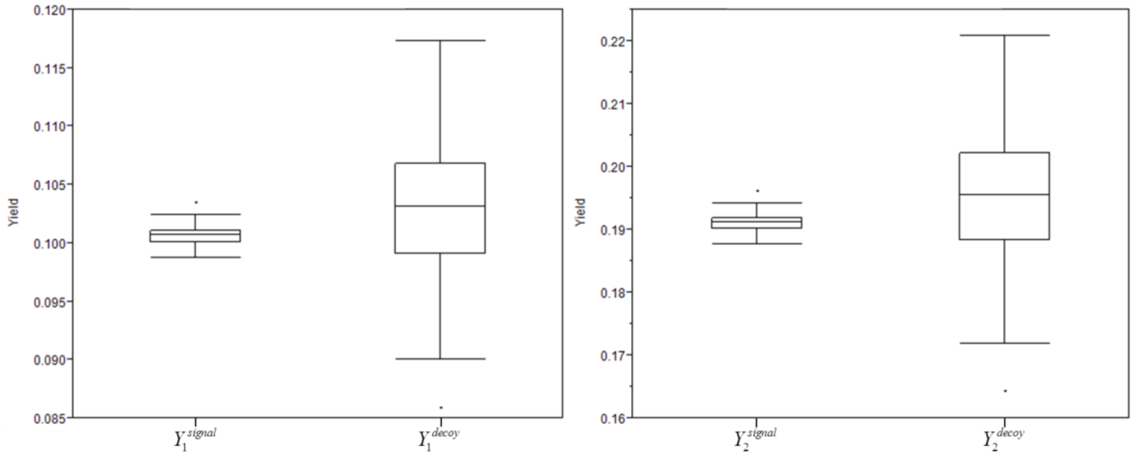


Figure 6. Example Photon Dependent Yields for *n* = 1 (left) and *n* = 2 (right). Note scales are different.

Table 9. Example Photon Number Dependent Yield Statistics.

State	Parameter	Estimate	Lower 90% CI	Upper 90% CI
Y_1^{signal}	Mean	0.100606	0.100487	0.100725
	Std Dev	0.000786	0.000711	0.000881
Y_1^{decoy}	Mean	0.103142	0.102314	0.103969
	Std Dev	0.005470	0.004947	0.006128
Y_2^{signal}	Mean	0.191010	0.190796	0.191224
	Std Dev	0.001414	0.001279	0.001584
Y_2^{decoy}	Mean	0.195536	0.194051	0.197021
	Std Dev	0.009813	0.008875	0.010993

1) Implementation Example

Prepare and measure QKD implementations often send qubits in frames, where a classical reference pulse is followed by a number of quantum-level pulses (e.g., a reference pulse followed by 1,000 weak coherent pulses). When studying our simulation results, we realized calculating the sent pulses according to frame sent was causing undo distortions in our measured gains Q_μ and Q_μ. For example, while the number of sifted signal detections was accurate, the number of sent signals was generally reported with an overage because of the transmission frame structure. While the reported difference may be small when compared to the number of pulse sent (e.g., 100s compared to 100,000s), when differentiating photon specific interference with significant losses, precision matters greatly.

Accuracy in the number of signal, decoy, vacuum pulses sent, received, and sifted is critical for calculating the necessary signal and decoy state gains used to estimate the photon number dependent yields Y_n. Poor precision of these measured values leads to poor bounds on security condition tolerances, potentially preventing decoy state QKD systems for detecting minute changes in photon specific interference. Furthermore, the number of corrected errors during error reconciliation is critical for calculating the necessary signal and decoy state QBERs, and resulting photon number dependent QBERs e_n. Likewise, accuracy of a system's dark count Y_0 is necessary for calculating the efficiency and photon number yields as identified in Eqs. (24), (25), (29), and (30). Achieving confidence in this low probability error (i.e., 1.0E-6) is gained through large sample size (e.g., >> 1 million vacuum states) as described in Eq. (11). Precision of these values is necessary for successfully implementing the decoy state security condition.

Lastly, this example provides insights into the number of vacuum and decoy detections necessary to provide accurate measurements for implementing the decoy state protocol. The inherent limitation with only collecting enough signal detections for 10k error reconciliation block sizes severely limits the accuracy of the measured decoy state gain and QBER, as well as, the number of system dark counts. For example, taking into considering the low probability of dark counts (e.g., 1.0E-6) and high transmission losses (e.g., >99%), large numbers of sent pulses (e.g., >500 million or more) may be necessary to achieve the desired statistical confidence for dark counts.

2) Implementation Considerations

When considering implementation of the decoy state protocol, one should first consider decoy states have been primarily used to increase secret key rates and not necessarily for the security condition. There has been little formal consideration of the effectiveness of the decoy state security condition in detecting an adversary in published literature. Therefore, we provide an enumerated list of practical implementation considerations and operational configuration issues which merit further examination for the detection of an adversary using the decoy state protocol:

1. Accuracy in signal and decoy state MPNs

2. Difference between signal and decoy state MPNs
3. Ratio of signal, decoy, and vacuum occurrence percentages
4. The impact of losses through the end-to-end quantum path
5. Induced errors between Alice and Bob, including reference alignment, timing synchronization, and bias in photon detection
6. Statistical confidence of decoy state security related parameters
7. Monitoring of alternate decoy state security measurable parameters such as the ratio between Q_μ and Q_ν as suggested in [52] or η for the signal and decoy state
8. Trade space definition between system performance (i.e., secret key rate) and security posture (i.e., detectability of an eavesdropper)

Lastly, we suggest conventional security practices such as operating from a known secure state and continuous monitoring be integrated into QKD system design and operation. For example, an independent characterization of channel losses, Bob's internal losses, detector efficiency, and protocol efficiency can be used as an additional security check for validating and ensuring expected yields during calibration or re-calibration of QKD systems during initialization, operation, and periodic security checks.

VI. Conclusions

This paper provides a detailed discussion of a modeled decoy state enabled QKD system. We also demonstrate verification activities at the component level and sub-system, as well as, system-level validation proving the model's correctness and applicability for further performance and security analysis. Lastly, based on our work modeling of a decoy state enabled QKD system, we bring attention to practical considerations for executing the decoy state protocol security condition, including design considerations and allowable tolerances.

Future work includes studying the decoy state enabled QKD system's ability to detect photon number specific attacks, accomplishing sensitivity analysis of operational parameters, conducting system-level performance characterization with respect to decoy state protocol security condition, and defining the security-performance trade space.

VII. Acknowledgements

This work was supported by the Laboratory for Telecommunication Sciences [grant number 5743400-304-6448].

VIII. Disclaimer

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the U.S. Government.

References

[1] L. O. Mailloux, M. R. Grimaila, D. D. Hodson, G. Baumgartner and C. McLaughlin, "Performance evaluations of quantum key distribution system architectures," *IEEE Security and Privacy,* vol. 13, no. 1, pp. XX-XX, 2015.

[2] J. D. Morris, M. R. Grimaila, D. D. Hodson, C. V. McLaughlin and D. R. Jacques, "Using the discrete event system specification to model quantum key distribution system components," *Journal of Defense Modeling and Simulation,* p. 1548512914554404, 2014.

[3] N. Gisin, G. Ribordy, W. Tittel and H. Zbinden, "Quantum cryptography," *Reviews of Modern Physics,* vol. 74, no. 1, pp. 145-195, 2002.

[4] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus and M. Peev, "The security of practical quantum key distribution," *Reviews of modern physics,* vol. 81, no. 3, pp. 1301-1350, 2009.

[5] S. Loepp and W. K. Wooters, Protecting Information, New York: Cambridge University Press, 2006.

[6] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2010.

[7] Office of Naval Research, "Quantum Information Sciences Program," U.S. Navy, [Online]. Available: http://www.onr.navy.mil/en/Science-Technology/Departments/Code-31/All-Programs/311-Mathematics-Computers-Research/Quantum-Information-Science.aspx. [Accessed 23 October 2014].

[8] Department of the Army, "U.S. Army Research Laboratory (ARL) Center for Distributed Quantum Information (CDQI)," [Online]. Available: https://www.fbo.gov/index?s=opportunity&mode=form&tab=core&id=0ae2481dfa71231c782f6e03ace76195.

[Accessed 23 October 2014].

[9] U. S. Air Force Scientific Advisory Board, "Utility of Quantum Systems for the Air Force," [Online]. Available: http://www.sab.af.mil/library/factsheets/factsheet.asp?id=21756. [Accessed 23 October 2014].

[10] ID Quantique SA, "Main page," [Online]. Available: www.idquantique.com. [Accessed 30 September 2014].

[11] SeQureNet, [Online]. Available: www.sequrenet.com. [Accessed 30 September 2014].

[12] Quintessence Labs, [Online]. Available: www.quintessencelabs.com. [Accessed 30 September 2014].

[13] MagiQ Technologies, [Online]. Available: www.magiqtech.com. [Accessed 30 September 2014].

[14] Quantum Communication Technology Co., Ltd., [Online]. Available: http://www.quantum-info.com/en.php. [Accessed 30 September 2014].

[15] S. Wiesner, "Conjugate coding," *ACM Sigact News,* vol. 15, no. 1, pp. 78-88, 1983.

[16] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, 1984.

[17] G. S. Vernam, "Cipher printing telegraph systems for secret wire and radio telegraphic communications," *American Institute of Electrical Engineers, Transactions of the,* vol. 45, pp. 295-301, 1926.

[18] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal,* vol. 28, pp. 656-715, 1949.

[19] R. Renner, N. Gisin and B. Kraus, "An information-theoretic security proof for QKD protocols," *Physical Review A,* vol. 72, no. 1, p. 012332, 2005.

[20] D. Gottesman, H.-K. Lo, N. Lutkenhaus and J. Preskill, "Security of quantum key distribution with imperfect devices," in *In Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, 2004.

[21] V. Scarani and C. Kurtsiefer, "The black paper of quantum cryptography: real implementation problems," *arXiv:0906.4547v2,* 2009.

[22] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail and J. Smolin, "Experimental quantum cryptography," *Journal of Cryptology,* vol. 5, no. 1, pp. 3-28, 1992.

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

[24] G. Brassard, N. Lutkenhaus, T. Mor and B. C. Sanders, "Limitations on Practical Quantum Cryptography," *Physical Review Letters,* vol. 85, no. 6, p. 1330, 2000.

[25] N. Lütkenhaus, "Security against individual attacks for realistic quantum key distribution," *Physical Review A ,* vol. 61, no. 5, p. 052304., 2000.

[26] W.-Y. Hwang, "Quantum key distribution with high loss: Toward global secure communication," *Physical Review Letters ,* vol. 91, no. 5, p. 057901, 2003.

[27] H.-K. Lo, X. Ma and K. Chen, "Decoy state quantum key distribution," *Physical Review Letters,* vol. 94, no. 3, p. 230504, 2005.

[28] X.-B. Wang, "Beating the photon-number-splitting attack in practical quantum cryptography," *Physical review letters,* vol. 94, no. 23, p. 230503., 2005.

[29] X. Ma, B. Qi, Y. Zhao and H.-K. Lo, "Practical decoy state for quantum key distribution," *Physical Review,* vol. 72, no. 1, p. 012326, 2005.

[30] X.-B. Wang, "Decoy-state protocol for quantum cryptography with four different intensities of coherent light," *Physical Review A,* vol. 72, no. 1, p. 012322, 2005.

[31] Y. Zhao, B. Qi, X. Ma, H.-K. Lo and L. Qian, "Experimental quantum key distribution with decoy states," *Physical review letters,* vol. 96, no. 7, p. 070502, 2006.

[32] Y. Zhao, B. Qi, X. Ma, H.-K. Lo and L. Qian, "Simulation and implementation of decoy state quantum key distribution over 60km telecom fiber," in *Information Theory, 2006 IEEE International Symposium on*, 2006.

[33] X.-B. Wang, C.-Z. Peng and J.-W. Pan, "Simple protocol for secure decoy-state quantum key distribution with a loosely controlled source," *Applied physics letters,* vol. 90, no. 3, p. 031110, 2007.

[34] X.-B. Wang, L. Yang, C.-Z. Peng and J.-W. Pan, "Decoy-state quantum key distribution with both source errors and statistical fluctuations," *New Journal of Physics,* vol. 11, no. 7, p. 075006, 2009.

[35] J.-Z. Hu and X.-B. Wang, "Reexamination of the decoy-state quantum key distribution with an unstable source," *Physical Review A,* vol. 82, no. 1, p. 012331, 2010.

[36] J.-Z. Hu and X.-B. Wang, "Secure quantum key distribution in an easy way," *arXiv*, p. arXiv:1004.3730, 2010.

[37] J. W. Harrington, J. M. Ettinger, R. J. Hughes and J. E. Nordholt, "Enhancing practical security of quantum key

distribution with a few decoy states," *arXiv*, pp. quant-ph/0503002, 2005.

[38] T. Horikiri and T. Kobayashi, "Decoy state quantum key distribution with a photon number resolved heralded single photon source," vol. 73, no. 3, 2006.

[39] M. Hayashi, "General theory for decoy-state quantum key distribution with an arbitrary number of intensities," *New Journal of Physics,* vol. 9, no. 8, 2007.

[40] W. Mauerer and C. Silberhorn, "Quantum key distribution with passive decoy state selection," *Physical Review A,* vol. 75, no. 5, p. 050305, 2007.

[41] G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," *In advances in Cryptology—EUROCRYPT'93,* pp. 410-423, 1994.

[42] L. Lydersen, N. Jain, C. Wittmann, Ø. Marøy, J. Skaar, C. Marquardt, V. Makarov and G. Leuchs, "Superlinear threshold detectors in quantum cryptography," *Phys. Rev.,* vol. A 84, no. 032320, 2011.

[43] L. O. Mailloux, M. R. Grimaila, D. D. Hodson and C. McLaughlin, "Modeling Continuous Time Optical Pulses in a Quantum Key Distribution Discrete Event Simulation," in *International Conference on Security and Management SAM'14*, 2014.

[44] D. C. Giancoli, Physics for Scientists and Engineers, Prentice Hall, 1989.

[45] D. A. Miller, Quantum mechanics for scientists and engineers, Cambridge University Press, 2008.

[46] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 2010 Winter Simulation Conference*, 2010.

[47] M. I. Smith, D. J. Murray-Smith and D. Hickman, "Verification and validation issues in a generic model of electro-optic sensor systems," *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* , vol. 4, no. 1, pp. 17-27, 2007.

[48] M. Roza, J. Voogd and D. Sebalj, "The Generic Methodology for Verification and Validation to support acceptance of models, simulations and data," *the Journal of Defense Modeling and Simulation: Applications, Methodology, Technology,* vol. 10, no. 4, pp. 347-365, 2012.

[49] T.-Y. Chen, H. Liang, Y. Liu, W.-Q. Cai, L. Ju, W.-Y. Liu, J. Wang, H. Yin, K. Chen, Z.-B. Chen, C.-Z. Peng and J.-W. Pan, "Field test of a practical secure communication network with decoy-state quantum cryptography," *Optics express* , vol. 17, no. 8, pp. 6540-6549, 2009.

[50] A. R. Dixon, Z. L. Yuan, J. F. Dynes, A. W. Sharpe and A. J. Shield, "Gigahertz decoy quantum key distribution with 1 Mbit/s secure key rate," *Optics express,* vol. 16, no. 23, pp. 18790-18979, 2008.

[51] J. F. Dynes, Z. L. Yuan, A. W. Sharpe and A. J. Shields, "Practical quantum key distribution over 60 hours at an optical fiber distance of 20km using weak and vacuum decoy pulses for enhanced security," *Optics express* , vol. 15, no. 13, pp. 8465-8471, 2007.

[52] Z. L. Yuan, A. W. Sharpe and A. J. Shields, "Unconditionally secure one-way quantum key distribution using decoy pulses," *Applied physics letters,* vol. 90, no. 1, pp. 011118-011118, 2007.

[53] A. R. Dixon, Z. L. Yuan, J. F. Dynes, A. W. Sharpe and A. J. Shields, "Continuous operation of high bit rate quantum key distribution," *Applied Physics Letters,* vol. 96, no. 16, p. 161102, 2010.

[54] C.-Z. Peng, J. Zhang, D. Yang, W.-B. Gao, H.-X. Ma, H. Yin, H.-P. Zeng, T. Yang, X.-B. Wang and J.-W. Pan, "Experimental long-distance decoy-state quantum key distribution based on polarization encoding," *Physical review letters,* vol. 98, no. 1 , p. 010505, 2007.

AFIT-ENV-MS-15-M-181

V. Journal Article II: Case Study: Applying Model-Based Systems Engineering to Quantum Key Distribution

This article describes the Model-Based Systems Engineering (MBSE) Processes, Methods, and Tools (PMTs) used to design, implement, verify and validate a Decoy State Enabled Quantum Key Distribution (QKD) model. We present a review of MBSE PMTs and a high-level discussion of QKD theory and practical implementation vulnerabilities. This article presents the first application of MBSE to design and analyze decoy state enabled QKD systems. Department of Defense (DoD) Architecture Framework (DoDAF) viewpoints are used to capture and communicate key model design considerations. Use cases are employed to define behaviors which are refined into measureable requirements that are assessed during Verification and Validation.

This paper was submitted to *IEEE Transactions on Systems, Man, and Cybernetics: Systems* on February 24, 2015 and is currently awaiting review.

Initial Submission: 24 February 2015
Version: 1.1

Developing a Decoy State Enabled Quantum Key Distribution System Model

Abstract: Quantum Key Distribution (QKD) is an innovative technology which exploits the laws of quantum mechanics to generate and distribute unconditionally secure cryptographic keys. This technology is suitable for use in applications where high levels of secrecy are required such as banking, government, and military environments. However, QKD technology is not well understood and practical implementations differ significantly from their theoretical design, which raises concerns about their use. In this paper, we use a Model-Based Systems Engineering (MBSE) approach to develop a QKD system model to study the impact of practical engineering limitations on system performance and security. Specifically, we detail the use of architectural products and use cases to identify, define, and analyze requirements for a decoy state enabled QKD system model. Using these requirements, we perform functional decomposition and implement a system-level model in a hardware-focused simulation. Additionally, we perform verification and validation activities to demonstrate the correctness and suitability of our model. Finally, the decoy state enabled QKD system model is demonstrated to successfully detect the photon number splitting attack.

Key Words: Quantum Key Distribution, Modeling & Simulation, Model-Based System Engineering

Authors Contact Information:

Ryan D. Engle, Student Member IEEE (Corresponding Author)
Master's Student in Systems Engineering
United States Air Force Institute of Technology, Wright-Patterson AFB, OH 45433-7765
Ryan.Engle@afit.edu
Telephone: 937-255-3636, Extension 7559

Michael R. Grimaila, Senior Member IEEE
Professor and Head, Department of Systems Engineering and Management
United States Air Force Institute of Technology, Wright-Patterson AFB, OH 45433-7765
Michael.Grimaila@afit.edu
Telephone: 937-255-3636, Extension 4800

Logan O. Mailloux, Student Member IEEE
PhD Candidate in Systems Engineering
United States Air Force Institute of Technology, Wright-Patterson AFB, OH 45433-7765
Logan.Mailloux@afit.edu, LoganMailloux@yahoo.com

Douglas D. Hodson
Assistant Professor of Computer Engineering
United States Air Force Institute of Technology, Wright-Patterson AFB, OH 45433-7765
Douglas.Hodson@afit.edu

Gerald Baumgartner
Research Physicist, Quantum Information Science
Laboratory for Telecommunication Sciences, College Park, MD 20740
Gbaumgartner@ltsnet.net

Colin V. McLaughlin
Research Physicist, Advanced Photonics
Naval Research Laboratory, Washington, DC 20375
Colin.Mclaughlin@nrl.navy.mil

I. INTRODUCTION

Quantum Key Distribution (QKD) is an innovative technology which exploits the laws of quantum mechanics to generate and distribute unconditionally secure cryptographic keys. QKD is unique in its ability to detect the presence of an eavesdropper attempting to subvert the distribution of secret key material and is suitable for use in applications where high levels of secrecy are required such as banking, government, and military environments. However, QKD technology is not well understood and practical implementations differ significantly from their theoretical design due to the non-ideal components used to build real-world systems [1]. There is a critical need to understand the impact of practical engineering limitations and implementation non-idealities on system performance and security.

In this paper, we use a Model-Based Systems Engineering (MBSE) approach to develop a system-level model to understand and study decoy state enabled QKD system implementations. An MBSE methodology is well suited to study the inherent complexities of QKD systems, as we seek to understand and study the impact of implementation non-idealities on system performance and security, and explore design and implementation variations across an assortment of hardware, software, and protocol configurations [2]. Furthermore, analysis of QKD systems requires expertise across multiple disciplines of expertise including electrical and computer engineering, computer science, information theory, optical communications, and quantum physics, which lends itself to the emerging MBSE methodology [3]. This research is focused on the use of MBSE to facilitate requirements definition, systems analysis, functional decomposition, protocol design, and testing activities in support of model development.

This article is organized as follows: Section II provides an overview of QKD theory and the decoy state protocol. Section III discusses the application of MBSE methodology to develop a decoy state QKD model using architectural framework products and use cases. In Section IV verification and validation (V&V) activities are presented to the correctness and suitability of our model. Finally, Section V presents conclusions and findings associate with our MBSE approach. This research is the first known application of MBSE to define and analyze a decoy state enabled QKD system. Appendix A contains the complete set of architectural artifacts created in this study. Appendix B provides a selection of use cases of primary interest for studying decoy state enabled QKD system. Lastly, Appendix C details the full set of requirements and associated V&V methods.

II. BACKGROUND

In this section, we provide a brief introduction to QKD including fundamental theory, unconditional security, practical implementation limitations, and the decoy state protocol.

A. Quantum Key Distribution Theory

The genesis of QKD can be traced back to Stephen Wiesner, who first developed the idea of encoding messages on photons in the late 1960s [4]. He proposed the idea of polarizing photons in conjugate bases to securely communicate information as quantum bits, called "qubits" [5]. In 1984, Charles Bennett and Gilles Brassard expanded this idea and proposed the first QKD protocol, known as "BB84", to securely distribute cryptographic key between two parties [6].

Table 1 illustrates an example of the BB84 polarization-based prepare and measure QKD protocol, where the sender "Alice" prepares qubits in one of four polarization states, |↔⟩, |↕⟩, |⤢⟩, or |⤡⟩, according to a randomly selected basis (i.e., rectilinear "⨁" or diagonal "⨂") and bit value (i.e., 0 or 1). These qubits are sent over a fiber optic "quantum channel" to the receiver "Bob" where they are measured using a randomly selected basis (i.e., ⨁ or ⨂) selected by Bob. In this manner, if Bob measures the qubits in the same basis in which Alice prepared them, the encoded message is obtained with a high degree of accuracy. Alternatively, if he measures the qubits in opposite (i.e., conjugate) basis, a random result occurs as depicted by the grey cells in Table 1. This unique phenomenon is due to the fundamental properties of quantum mechanics where measuring single photons disturbs the quantum system [7]. This process of preparing, sending, and measuring qubits is known as “quantum exchange” and results in raw keys at both Alice and Bob.

Table 1. Polarization-Based Prepare and Measure Example.

The Sender “Alice” Prepares			**The Receiver “Bob” Measures**	
Random Bit Value	**Random Basis**	**Prepared Polarization State**	**Random Basis**	**Measured Bit Value**
0	⊕	|↔⟩	⊕	0
0	⊕	|↔⟩	⊗	Random 0 or 1
1	⊕	|↕⟩	⊕	1
1	⊕	|↕⟩	⊗	Random 0 or 1
0	⊗	|⤡⟩	⊕	Random 0 or 1
0	⊗	|⤡⟩	⊗	0
1	⊗	|⤢⟩	⊕	Random 0 or 1
1	⊗	|⤢⟩	⊗	1

Following successful quantum exchange, the raw keys are sifted to eliminate mismatched basis measurements. Sifting is accomplished by announcing basis information for each qubit over a conventional networked communication medium, i.e., a “classical channel” where only the bases are broadcast and not the encoded bit values. This results in a shared sifted key between Alice and Bob approximately half the length of the original raw key due to Bob’s random basis selection (i.e., ideally, 50% rectilinear ⊕ and 50% diagonal ⊗). Next, error reconciliation is performed to correct errors in the sifted keys using specialized bi-directional error correction algorithms (e.g., Winnow, Cascade, or Low-Density Parity-Check) [8].

The error reconciled key is then subject to privacy amplification – an advanced information theory technique used to ensure an eavesdropper “Eve” has negligible information regarding the key [8]. This results in a smaller and more secure key. Finally, Alice and Bob confirm they have matching keys by comparing hashes of their respective keys. If the hashes match, the final secret key is delivered. The QKD-generated shared secret key can be used to increase the security of conventional symmetric encryption algorithms such as DES, 3DES, or AES through frequent re-keying. Alternatively, QKD-generated key is often described as an enabler for the unbreakable One-Time Pad (OTP) encryption algorithm [9], [10]. More detailed descriptions of these processes are available in [1], [7], [8].

B. QKD’s Unconditional Security

The security of QKD is based on quantum uncertainty, where directly measuring a qubit changes its quantum state [7]. A classical communication bit exists in a deterministic state of either 0 or 1 where it can be measured and re-measured as necessary without disturbing the state. In contrast, a qubit exists in a probabilistic superposition of states and cannot be successfully copied, cloned, or amplified without disturbing the originally encoded state [11]. Specifically, any measurement of the qubit will cause it to collapse into one of the measurement basis states. By preparing qubits in conjugate bases (i.e., ⊕ and ⊗) as prescribed in BB84, eavesdropping on the quantum channel necessarily increases the system’s Quantum Bit Error Rate (QBER) since the listener does not know Alice’s randomly selected basis. [12].

In QKD security proofs, all quantum bit errors are attributed to an eavesdropper, since the source of errors cannot be known [8]. If a defined QBER threshold is exceeded, the key generation process is aborted, as it is assumed an adversary is interfering with the quantum key exchange. Thus, by closely monitoring the QBER, Alice and Bob are able to determine if an eavesdropper is listening to the key generation process. QKD's unconditional security claim is founded upon the system's QBER as captured in formal proofs [12], [13].

C. Practical Implementation Limitations

Despite unconditional security claims, real-world QKD systems are constructed from non-ideal components with practical engineering limitations which violate critical assumptions used in formal security proofs [14]. For example, perfect on-demand single photon sources are assumed to generate qubits. However, perfect on-demand single photon sources cannot be realized using current technology. Instead, classical laser sources are used to produce "strong" optical pulses (i.e., pulses with millions of photons), which are attenuated down to "weak coherent pulses" (i.e., pulses with a Mean Photon Number (MPN) of < 1 photon per pulse). These sub-quantum energy levels are probabilistically represented using the Poisson distribution $P(n|\mu) = \frac{\lambda^n e^{-\mu}}{n!}$, where n is the number of photons in the pulse and μ = MPN. This means, for example, when the MPN = 0.1, 90.48% of the pulses have no photons, 9.05% have one photon, and 0.47% have two or more photons.

While low MPNs attempt to retain the theoretical security requirements of QKD at the expense of the key generation rate, attenuated laser sources unavoidably result in multiphoton pulses, each of which exposes information about the secret key to a listening adversary. This multiphoton vulnerability has been subject of a number of attacks against QKD systems and specifically the Photon Number Splitting (PNS) attack [15], [16], [17], [18]. This attack allows an all-powerful eavesdropper, Eve to generate the same secret key as Alice and Bob without making her presence known to the authorized participants. In this attack, Eve blocks all single photons pulses and steals a photon from multi-photon pulses as they propagate through the quantum channel. In this attack, Eve can measure the stolen photon without affecting the pulse that propagates to Bob.

D. The Decoy State Protocol

In order to mitigate the PNS attack, the decoy state protocol was introduced in 2003 by Hwang [19] and further refined by Lo *et. al* [20]. Table 2 describes each of the three decoy state pulse types as suggested by Ma *et. al* [21], where each has a unique MPN and an accompanying occurrence percentage. In the decoy state protocol, signal, decoy, and vacuum pulses are randomly selected by Alice for transmission over the quantum channel, and their type is indistinguishable to eavesdroppers. Only during sifting is Bob made aware of the pulse type for his detections. While the signal states (i.e., pulses) are used for generating secret key, decoy states are used for detecting a PNS attack. Specifically, Alice and Bob compare photon yield and error rate statistics between the signal and decoy states to detect the presence of an adversary perpetrating a PNS attack.

Table 2. Decoy State Protocol Configuration as Reported in [22].

State	State Description	MPN	Occurrence Percentage
Signal	The signal state is used to transmit quantum pulses for generating shared secret key and facilitates higher key distribution rates through an increased MPN (e.g., an MPN of 0.6 is higher than 0.1)	0.65	75%
Decoy	The decoy state increases the likelihood of detecting the PNS attack, specifically photon number dependent interference, on the quantum channel through statistical differentiation with the signal state.	0.08	12.5%
Vacuum	The vacuum state is used to determine the detector dark count rate (i.e., noise) when no photons are present.	~0	12.5%

While the decoy state protocol has been proposed as a means to mitigate the PNS attack, it has not been well studied in realized QKD systems nor has the protocol been defined in a way which enables consistent and efficient implementation. As a consequence many design decisions are left to system developers without proper understanding of their impact on system performance and security. Given the complex, multi-disciplinary nature of QKD systems, we have chosen to employ an MBSE approach to design, build, and test a decoy state simulation-based QKD system model to further understand this increasingly popular attack mitigation technology.

III. APPLICATION OF MODEL-BASED SYSTEMS ENGINEERING TO DEVELOP THE DECOY STATE ENABLED QKD SYSTEM MODEL

MBSE is the application of modeling to systems engineering processes, methods, and tools [2]. MBSE provides an organized, repeatable, iterative, and convergent approach to understand, communicate and meet, user needs throughout the development of complex, interdisciplinary systems [23], [24]. More specifically, MBSE is a model-centric approach that supports requirements definition, system design, analysis, and V&V activities whose main deliverables are a model and supporting artifacts which facilitate shared understanding and clear communication amongst stakeholders [25]. As an emerging methodology, MBSE is being investigated as an effective tool for understanding increasingly complex systems [3], [26], [27], [28]. For these reasons, MBSE is well suited to study QKD systems.

In this work, we interpret a “model” as an abstraction for the purpose of studying a QKD system of interest [29]; this understanding is consistent with the wider interpretation of MBSE [2], [3]. Thus, we use Department of Defense Architectural Framework (DoDAF) products [30], use cases [31, 32, 33], [32], [33], and other tools as “models” to successfully understand, communicate, and meet user needs. Such architecture viewpoints may be used to capture characteristics of an existing, i.e., “As-Is”, system and those of a desired, i.e., “To-Be,” system. These systems engineering models provide requirements traceability and aid in the definition, decomposition, implementation, integration, and V&V to deliver the desired decoy state enabled QKD system model. Additionally, the desired model can itself be used as an MBSE tool to aid system developers and stakeholders in defining, communicating, and measuring system performance and security.

A. Requirements Definition

MBSE is a requirements-driven approach; as such our first step begins with capturing stakeholder needs. In general, our stakeholders are interested in studying realized QKD systems, where the impact of practical engineering limitations and non-idealities on system performance and security is not well understood [34]. More specifically, they are interested in characterizing the performance of the decoy state protocol in realized QKD systems. To further understand and clearly communicate this need, we developed a tailored concept of operations (CONOPS) captured in a user’s manual [35], [36], [37], [38]. In our CONOPS we focused on capturing scope and context, focusing on the system of interest’s activities, relevant actors, boundary, and operational environment(s).

Fig. 1 illustrates the desired operational context, captured in a DoDAF Operational Viewpoint (OV-1) [39]. This figure provides a view of the QKD system which consists of Alice, Bob, Eve, a quantum channel, and a classical channel. Operating under normal conditions, Alice and Bob perform quantum exchange over the quantum channel, while the classical channel is used to facilitate the QKD protocol. The QKD system generates shared secret key, *K,* which is used by bulk encryptors to encrypt and decrypt message traffic (e.g., voice, data, video, etc.). During a PNS attack, Eve obtains the same shared secret key *K* as Alice and Bob and is then able to decrypt the encrypted communications. It is important to note that Eve's actions are not considered part of the QKD system of interest; however, she is necessary to validate stakeholder needs and verify decoy state protocol performance.

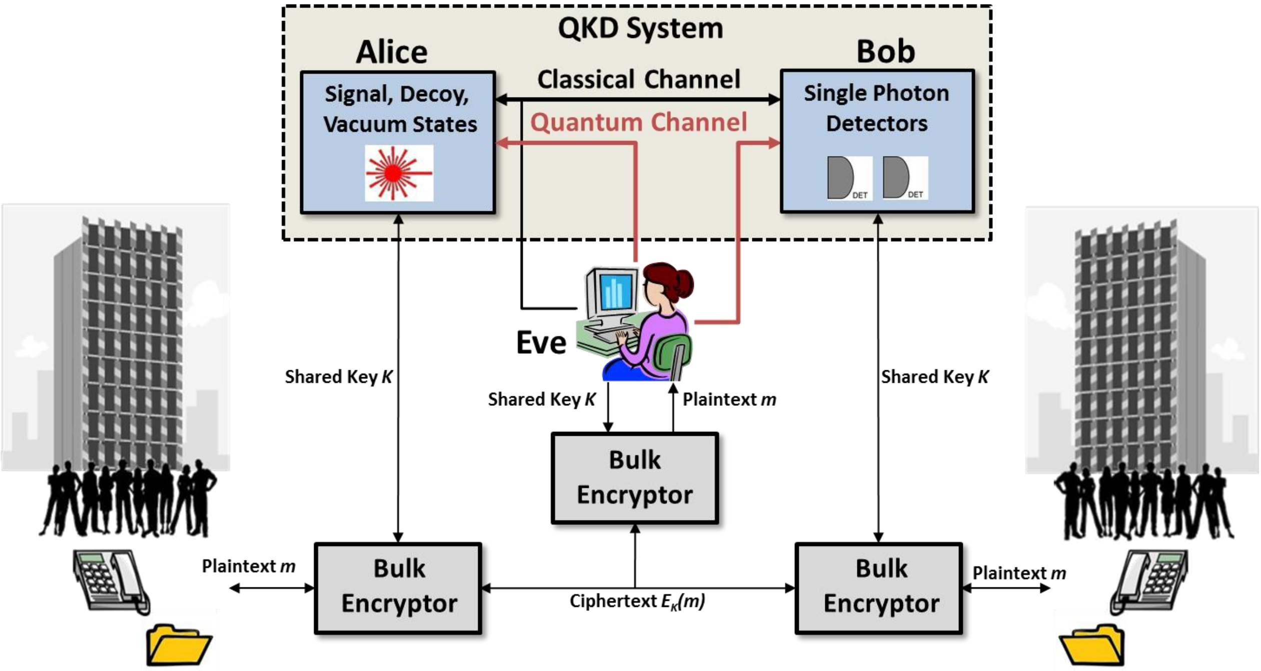


Fig. 1. A decoy state enabled QKD system is shown with an eavesdropper "Eve" listening to the secret key generation between Alice and Bob. Eve is able to obtain the secret key K and read the encrypted message traffic.

Table 3 provides an initial set of system requirements defined from the specified user needs and desired system capability. The initial requirements focused on defining the fundamental behaviors necessary to model a decoy state enabled QKD system. Although discussed sequentially, we realize requirements definition is an ongoing process where increased understanding gained through design and decomposition processes will lead to further refined requirements. Thus, using the MBSE methodology, we attempted to refine this initial set into refined requirements which were correct, unambiguous, verifiable, and traceable [31].Additionally, we identified V&V methods for each requirement according to best practices [40]. The V&V methods will be discussed further in Section IV.

Table 3. Initial Requirements.

Requirement	V&V Method
1. The model shall employ the Bennett and Brassard (BB84) protocol [6] [7].	Demonstration
2. The model shall extend the BB84 protocol with the decoy state protocol [19] [20] [21].	Inspection, Demonstration
3. The model shall detect an eavesdropper performing a PNS attack on the quantum channel.	Test
4. The model shall represent commercially available components when available [34].	Inspection, Demonstration

B. Protocol Design and Decomposition

Once the initial requirements and the operational context were defined, we decomposed the BB84 and decoy state protocols into functional activities. Further, we integrated the BB84 protocol and decoy state theory into a single protocol. We selected the DoDAF Operational Activity Decomposition Tree (OV-5a) [41] to facilitate this process and communicate our conceptual design amongst our stakeholders. Additionally, we selected the Systems Modeling Language (SysML) as an intuitive means to construct an accurate representation of our conceptual design [42].

Fig. 2 illustrates the resulting decoy state enabled BB84 protocol, where the first level of decomposition comprises seven operational activities. This decomposition is based on the logical integration of BB84's established phases of operation [6], [7], published decoy state theory [20], [21], practical implementation considerations [34], [43] and Subject Matter Expert (SME) inputs.[1] In the activity decomposition, BB84 activities are represented with tan blocks, while the decoy state activities are shown in green. The activities of interest in our model are: *Exchange Quantum Bit*, *Generate Sifted Key*, and *Detect Eavesdropper*. These activities are decomposed to the level of detail necessary to identify pertinent behaviors of interest for our study.

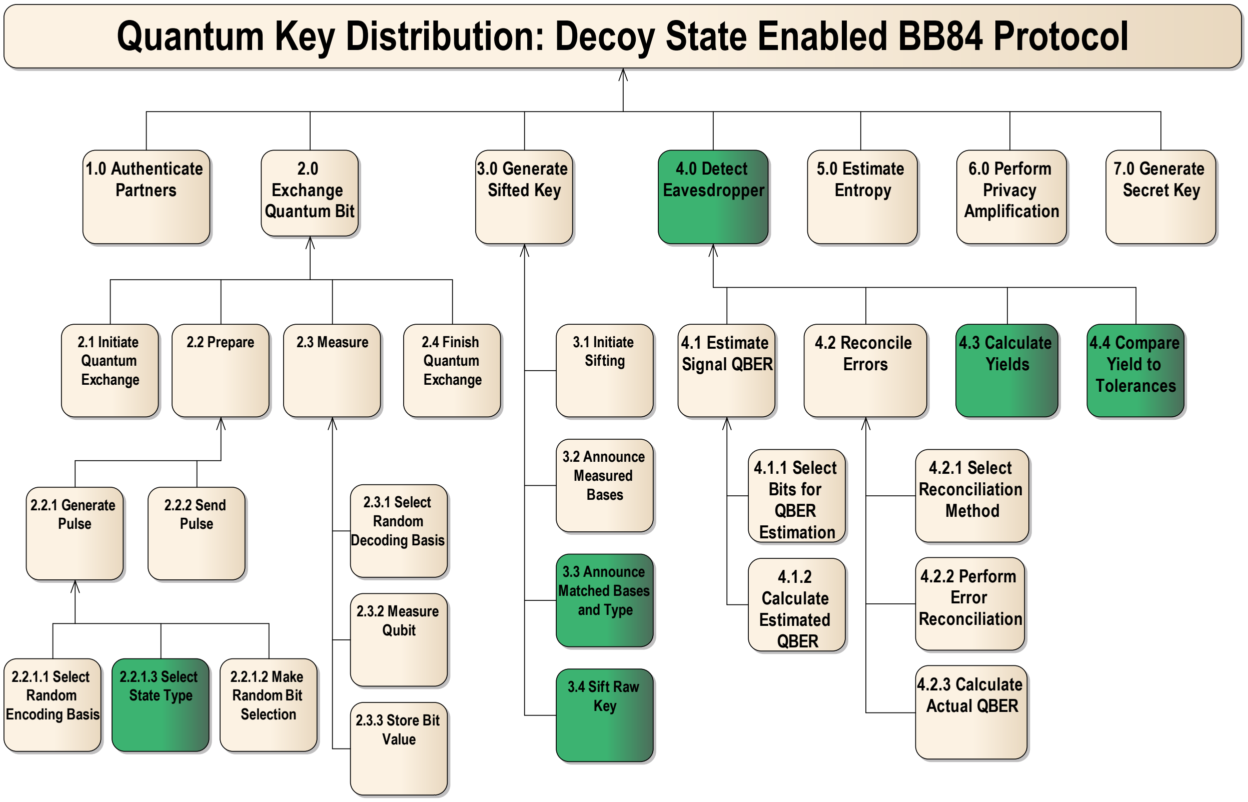

Quantum Key Distribution: Decoy State Enabled BB84 Protocol
1.0 Authenticate Partners
2.0 Exchange Quantum Bit
3.0 Generate Sifted Key
4.0 Detect Eavesdropper
5.0 Estimate Entropy
6.0 Perform Privacy Amplification
7.0 Generate Secret Key
2.1 Initiate Quantum Exchange
2.2 Prepare
2.3 Measure
2.4 Finish Quantum Exchange
3.1 Initiate Sifting
4.1 Estimate Signal QBER
4.2 Reconcile Errors
4.3 Calculate Yields
4.4 Compare Yield to Tolerances
2.2.1 Generate Pulse
2.2.2 Send Pulse
2.3.1 Select Random Decoding Basis
3.2 Announce Measured Bases
4.1.1 Select Bits for QBER Estimation
4.2.1 Select Reconciliation Method
2.3.2 Measure Qubit
3.3 Announce Matched Bases and Type
4.1.2 Calculate Estimated QBER
4.2.2 Perform Error Reconciliation
2.2.1.1 Select Random Encoding Basis
2.2.1.3 Select State Type
2.2.1.2 Make Random Bit Selection
2.3.3 Store Bit Value
3.4 Sift Raw Key
4.2.3 Calculate Actual QBER

Fig. 2. Decoy State Enabled BB84 Activity Decomposition. Note the activities are named with a descriptive verb phrase in conformance with the System Model Language, and not the commonly accepted physics nomenclature.

In this protocol, Alice and Bob first authenticate their identities to each other. Additionally, transactional authentication may be accomplished after each main activity for extra assurance of the shared secret key [8]. Next, *Exchange Quantum Bit* (i.e., encoded pulses – see Table 1) occurs until Bob has enough raw key bits to begin sifting. Specific to the decoy state protocol, each pulse has a randomly selected state type (i.e., signal, decoy, vacuum) defined by its MPN. *Generate Sifted Key* removes mismatched detections from Alice and Bob's raw key. Additionally, the decoy state protocol requires that

[1] Note: while the decoy state protocol addresses comparing signal and decoy state yields and error rates, we are only addressing yields in this protocol as an all-powerful Eve would not introduce errors.

the state type be announced and the Generate Sifted Key activity to be modified to account for the decoy and vacuum state types. The *Detect Eavesdropper* activity includes conventional BB84 activities to estimate and reconcile errors in the shared sifted keys and then checks for the PNS attack by comparing the calculated signal and decoy state yields.

C. Model Development through Use Cases

After identifying the activities and behaviors of interest, our model development effort transitioned to organizing, defining, and formalizing the relationships between associated protocol actors, behaviors, and resources to communicate the desired system capability into detailed use cases. This method also includes the identification of actors i.e., external entities that initiate actions, and the protocol boundary. First, we identified the actors – external entities (i.e., Alice and Bob Protocol) that initiate actions in the decoy state enabled BB84 system model as depicted in Fig. 3. The SysML-compliant diagram, depicts nine use cases based on our identified activities and clearly illustrates the relationships between the actors, protocol boundary, and the identified use cases [42], [31].

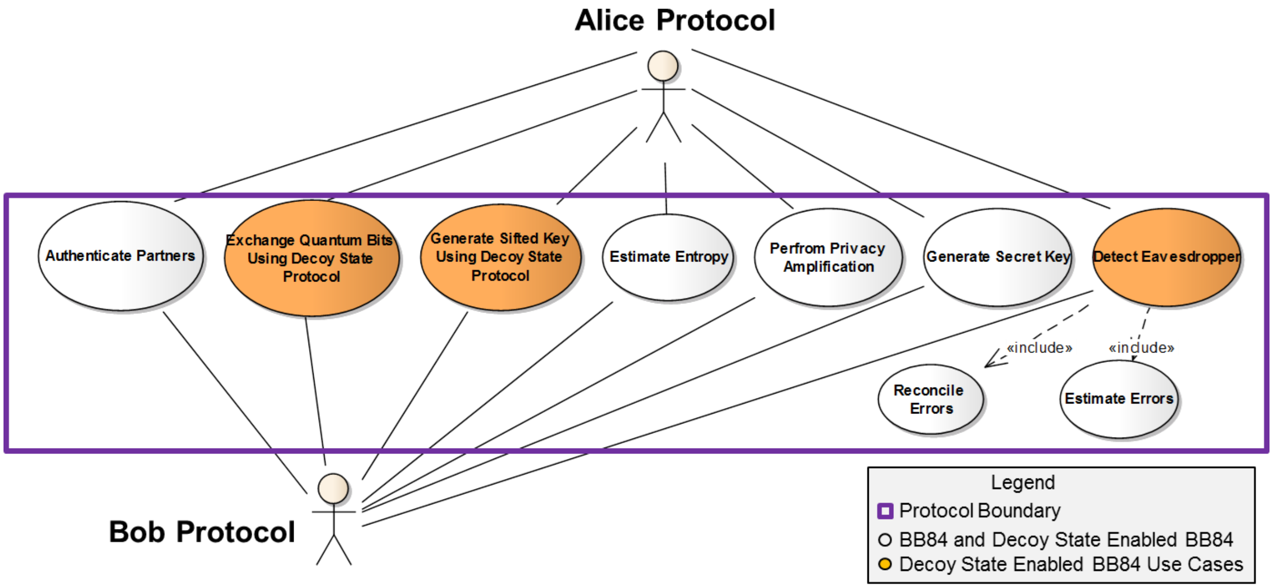

Alice Protocol
Authenticate Partners
Exchange Quantum Bits Using Decoy State Protocol
Generate Sifted Key Using Decoy State Protocol
Estimate Entropy
Perfrom Privacy Amplification
Generate Secret Key
Detect Eavesdropper
«include»
«include»
Reconcile Errors
Estimate Errors
Bob Protocol
Legend
▣ Protocol Boundary
○ BB84 and Decoy State Enabled BB84
● Decoy State Enabled BB84 Use Cases

Fig. 3. Decoy State Enabled BB84 Protocol Use Case Diagram.

To develop our decoy state enabled QKD system model, we are primarily interested in defining the behaviors associated with the *Exchange Quantum Bits…*, *Generate Sifted Key…*, and *Detect Eavesdropper* use cases. We created tailored use cases based on [31], [32], [33] to focus on the desired functional behavior defined in a main sequence, applicable algorithms, design decisions, definitions, and outstanding questions/issues. Initially, we created the three uses cases of interest from published literature references, and then subjected them to review by SMEs to verify relevant behaviors. These uses cases are provided in their entirety in Appendix B and were used to develop derived requirements for the desired system model.

Since the *Detect Eavesdropper* use case is of primary interest to our model it is included in its entirety. This use case was selected because the primary purpose of the decoy state protocol is to detect an eavesdropper conducting the PNS attack. This example represents one possible implementation of the decoy state protocol. Note: for readability the terms “Alice” and “Bob” were typically used instead of “Alice Protocol” and “Bob Protocol.”

Detect Eavesdropper Use Case

Summary: Alice and Bob calculate signal and decoy error rates, reconcile signal errors and attempt to detect an eavesdropper as well as detect a PNS attack. This step is unique to the decoy state enabled BB84 protocol and includes the *Error Estimation* and *Error Reconciliation* phases of BB84.

Dependency: Includes *Estimate QBER* (BB84 – Error Estimation) and *Reconciliation Signal Errors* (BB84 – Error Reconciliation).

Actor(s): Alice Protocol (Alice) and Bob Protocol (Bob).

Resources: System of interest.

Pre-condition: Partner's identities have been authenticated. Classical communication channel is established and authenticated. Calibration has occurred. Pre-established tolerances for the signal and decoy photon number dependent yield differences and a pre-established threshold for the estimated and actual signal and decoy QBERs are known.

Post-condition: No eavesdropper has been detected performing a PNS attack and the protocol is ready to continue.

Main sequence:

1. <<*Estimate Signal QBER*>>.
2. <<*Reconcile Signal Errors*>>.
3. Alice checks whether the actual decoy QBER exceeds the actual decoy QBER threshold.
4. Alice calculates and stores the measured gain by dividing the number of signal and decoy detections by the total number of signal and decoy pulses she sent to Bob.
$$Signal\ Gain, Q_\mu = \frac{\#\ of\ signal\ detections}{\#\ of\ signal\ pulses\ sent^*} \quad (1) \qquad Decoy\ Gain, Q_\nu = \frac{\#\ of\ decoy\ detections}{\#\ of\ decoy\ pulses\ sent^*} \quad (2)$$
5. Alice uses the measured signal and decoy gains to calculate and store the end to end efficiencies, η_{signal}, and η_{decoy}.
$$\eta_{signal} = \frac{-\ln\left|1+Y_0-Q_\mu\right|}{\mu} \quad (3) \qquad \eta_{decoy} = \frac{-\ln|1+Y_0-Q_\nu|}{\nu} \quad (4)$$
6. Alice uses the efficiencies to calculate and store the estimated 1-, 2-, 3-, 4-, and 5-photon yields for the signal and decoy states. The calculations for single (1-) photon yields, Y1, are shown below:
$$signal\ Y1 \cong Y_0 + \left[1-\left(1-\eta_{signal}\right)^1\right] \quad (5) \qquad decoy\ Y1 \cong Y_0 + \left[1-\left(1-\eta_{decoy}\right)^1\right] \quad (6)$$
7. Alice compares the photon yield of the signal to the 1-, 2-, 3-, 4-, and 5-photon yield decoy states and stores the results. The difference is within a pre-established tolerance. The calculation for single (1-) photon yield estimate difference is: $|signal\ Y1 - decoy\ Y1| \leq tolerance_1$ (7)
8. Alice does not detect a photon number splitting attack.

Alternative sequence:

7. Alice compares the single photon yield of the signal to the single photon yield decoy states. The difference is not within a pre-established tolerance.
8. Alice detects a photon number splitting attack, records the deviation and displays a warning.

Design Decisions:

- Definitions [21]:
  - Yield = the conditional probability of a detection event at Bob's side given that Alice sends out a pulse
  - Gain = product of the probability of Alice sending out a pulse (following a Poisson distribution) and the conditional probability that Alice's pulse (and background) will cause a detection at Bob
- Integrated Decoy State Error Test into this use case because it became a trivial (one line) use case.
- *That is, the number of pulses sent in the time frame during which Bob is ready to detect photons.
- The idea of tolerance is consistent with the notion that Alice and Bob must know their channel well [20].
- For simulation and certain analytical studies it is better to warn a user of a difference exceeding a tolerance rather than terminating the simulation. Strict termination would prevent data capture.

Outstanding questions/issues:

- NOTE: End to end efficiency is related to, protocol (e.g., 80% of signal states), channel (0.2 db/km), Bob's internal loss (2-5dB), and detector efficiency (10%) and dark counts.
- How could calibrated end-to-end efficiencies be employed in Step 5?
- When 2-photon, 3-photon, 4-photon, or n-photon yields are desired for comparison, the tolerance and difference for the n-photon yield must be calculated. For example:

$$signal\ Y_n \cong Y_0 + \left[1-\left(1-\eta_{signal}\right)^n\right] \quad (8) \qquad |signal\ Y_n - decoy\ Y_n| \leq tolerance_n \quad (9)$$

Defining the decoy state specific use cases resulted in a total of 108 functional and derived requirements available in Appendix C. At the time of definition one or more V&V methods were assigned to each requirement and were traceable through use cases during implementation. Identifying the verification method for these requirements at the time of their creation enabled the smooth application of V&V during incremental design-build-test cycles.

Table 4 illustrates traceability between requirements, the *Detect Eavesdropper* use case, and corresponding V&V method for a subset of derived requirements. Each requirement was written to conform to the quality attributes defined by [31]. These enumerated requirements are vital to the decoy state enabled QKD system model.

Table 4. Traceability Matrix for Decoy State Specific Requirements.

REQ #	Description	Use Case Reference(s)	V&V Method
2.1.5	Alice shall calculate and store the gain of the signal state qubits.	Detect Eavesdropper, Step 4	Demonstrate
2.1.6	Alice shall calculate and store the gain of the decoy state qubits.	Detect Eavesdropper, Step 4	Demonstrate
2.1.7	Alice shall calculate and store the end-to-end signal efficiency.	Detect Eavesdropper, Step 5	Demonstrate
2.1.8	Alice shall calculate and store the end-to-end decoy efficiency.	Detect Eavesdropper, Step 5	Demonstrate
2.1.9	Alice shall calculate and store the yield of the 1-photon signal pulses, i.e., the 1-photon signal yield.	Detect Eavesdropper, Step 6	Demonstrate
2.1.10	Alice shall calculate and store the yield of the 2-photon signal pulses, i.e., the 2-photon signal yield.	Detect Eavesdropper, Step 6	Demonstrate
2.1.14	Alice shall calculate and store the yield of the 1-photon decoy pulses, i.e., the 1-photon decoy yield.	Detect Eavesdropper, Step 6	Demonstrate
2.1.15	Alice shall calculate and store the yield of the 2-photon decoy pulses, i.e., the 2-photon decoy yield.	Detect Eavesdropper, Step 6	Demonstrate
2.1.19	Alice shall calculate and store the difference of the 1-photon signal yield and the 1-photon decoy yield.	Detect Eavesdropper, Step 7	Demonstrate
2.1.20	Alice shall calculate and store the difference of the 2-photon signal yield and the 2-photon decoy yield.	Detect Eavesdropper, Step 7	Demonstrate
3.1.1	The model shall detect an eavesdropper performing a photon number splitting (PNS) attack on the quantum channel by verifying that the difference of the 1-photon signal and decoy yield estimates is less than 0.00202 photons/pulse sent	Detect Eavesdropper, Step 7	Test
3.1.2	The model shall detect an eavesdropper performing a photon number splitting (PNS) attack on the quantum channel by verifying that the difference of the 2-photon signal and decoy yield estimates is less than 0.00399 photons/pulse sent	Detect Eavesdropper, Step 7	Test

The culmination of this design effort resulted in the conceptual QKD system model shown in Fig. 4. In the model, Alice and Bob contain a processor, which controls the protocol activities, and a Quantum Module (QM) which controls and executes the activities related to the exchange of quantum bits. Alice's quantum module contains a controller, which coordinates activities in the QM, a Classical Pulse Generator which generates the strong laser pulses, a Polarization Modulator, which adjusts the pulse polarization to encode basis and bit selections prepared by Alice, and a Decoy State Modulator which attenuates the pulses to the specified signal, decoy, or vacuum MPN. Bob's QM consists of complementary components to measure a random basis, via the Passive Basis Selector, and the bit values, via Single Photon Detectors.

In this conceptual model, the operational activities identified in Fig. 2, are allocated to each component and identified by number in Fig. 4; the decoy state protocol specific activities are underlined. In general, the electrical connections between components are shown as black bi-directional arrows and optical connections are shown in red. The classical channel is the only exception as it may be implemented across either electrical signal-based Ethernet or optical signal-based fiber cable.

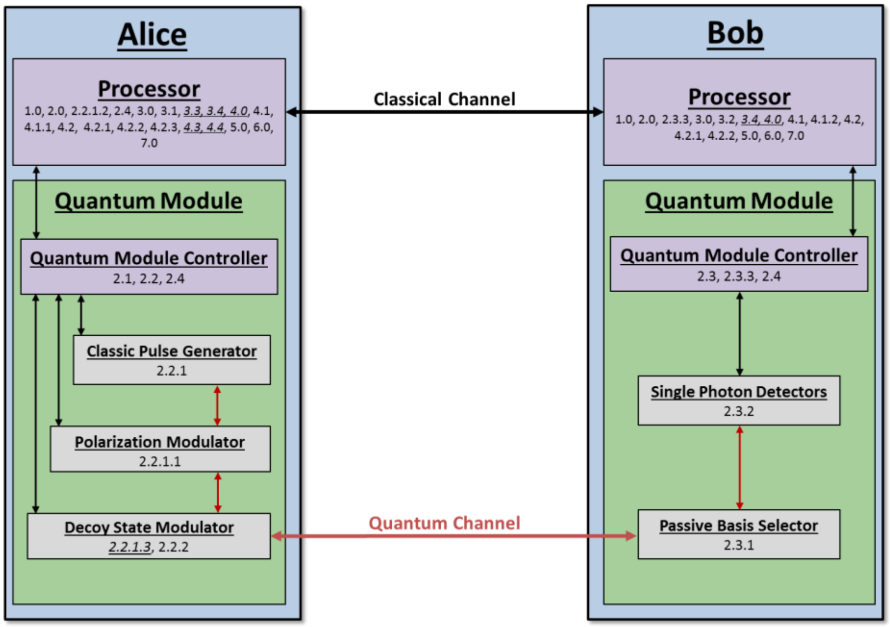


Fig. 4. Conceptual QKD Model and Components Mapped to Operational Activities

IV. VERIFICATION AND VALIDATION METHODOLOGY

Section III details the processes, methods, and tool used to define the detailed system design, leading to model implementation. Specifically, we used the functional decomposition to guide the design of the model and the use cases to capture the necessary system behaviors (i.e., akin to derived requirements). Furthermore, the use cases were treated as specifications for both implementation and V&V. In this section, V&V activities used to the correctness and suitability of our model are described.

A. Verification and Validation Traceability

Verification is a systems engineering process which confirms the system was built correctly, i.e., the system satisfies requirements [44], [45], [38]. In contrast, validation confirms the correct system was built or will be built [40]; modeling enables the "will be built" aspect of this concept [46], [47], [48], [29]. Traditionally, four methods are used to accomplish V&V: inspection, demonstration, testing, and analysis [40]. Inspection of the model was accomplished by comparing design artifacts to the model implementation, performing code review, and examining configuration files. Additionally we employed peer programming during the final stage of model development as an inspection tool. Demonstration typically involved the operation of the system model e.g., running an instance of the simulation. Testing assessed system conformance through controlled conditions and involved the utilization of scientific principles and procedures. Analysis applied analytical methods and tools, such as experimental design, to formally evaluate the system's compliance to stated requirements. V&V processes were performed incrementally throughout model development with increasing levels of detail.

B. Verification and Validation Results

First, we validated the system model against the experimental system reported in *Field Test of a Practical Secure Communication Network with Decoy-State Quantum Cryptography* by [22]. As practical QKD systems are dependent upon dozens of operational and implementation details (i.e., calibration of the quantum channel, programming of controllers, device imperfections, and the changing operational environment), we merely attempt to demonstrate sufficient accuracy for the intended purpose of studying the ability to detect Eve performing a PNS attack. This set of treatments resulted in a mean signal gain of 7.63E-3 which is reasonably close to the reported signal gain of 6.36E-3 (i.e., within the same order of magnitude).

Second, we defined photon number dependent yield tolerances to identify the PNS attack. Based on variance in the simulation, we determined 1000 simulations would be required to capture the upper bound of the tolerance [49]. Since the 2-, 3-, 4-, 5- photon yield difference share the same calculation dependencies, i.e. Eq. (3) and (4), as the 1-photon yield, we only performed this method for the once. After the simulations were performed, we found the tolerances for the photon number dependent yields as shown in Table 5.

Table 5. Photon Number Dependent Yield Tolerances

Photon Number Yield	Implemented Tolerance
Y1	0.00202
Y2	0.00399
Y3	0.00592
Y4	0.00780
Y5	0.00965

Next, the primary objective of the decoy state model is to detect a PNS attack; thus we designed an experiment to assess the model's ability to detect the PNS attack. Fig. 5a presents results from the experiment where unequal signal and decoy state yield estimates are shown indicating the PNS attack is detectable. Each pair of boxplots (Baseline and PNS) according to the photon number dependent yields (Y1, Y2, Y3, Y4, Y5) is shown with a clear difference between the Baseline and PNS results from 100 runs of the simulation. The differences increase as the photon number increases due to the power relationship as defined in the photon number dependent yield calculation identified in *Detect Eavesdropper* use case step 6. Specifically, as the photon number, n, increases, the quantity inside the parentheses is raised to the power of *n* as: $signal\ Y_n \cong Y_0 + \left[1 - \left(1 - \eta_{signal}\right)^n\right]$

During the baseline treatments, this causes both the signal and decoy yields to grow proportionally as the photon number increases resulting in a boxplot which appears to grow outward from a mean. In contrast, this difference becomes unbalanced during the PNS attack due to the difference in MPNs of the signal and decoy states, i.e. Eve will block more of the decoy state pulses which affects the decoy state yields more adversely due to its lower MPN. This means the signal state yields increase while the decoy state yields remain low resulting in a growing difference as shown in Fig. 5a.

Fig. 5b presents detailed simulation results for the Baseline and PNS configurations for the single photon yields, Y1. Black and green points depict similar performance during normal operations (i.e., without Eve), whereas red and blue illustrate the difference of yields during Eve's PNS attack. Again the difference between the Baseline and PNS results is apparent: the green and black Baseline points are approximately equal to each other, whereas the red and blue PNS points are clearly different.

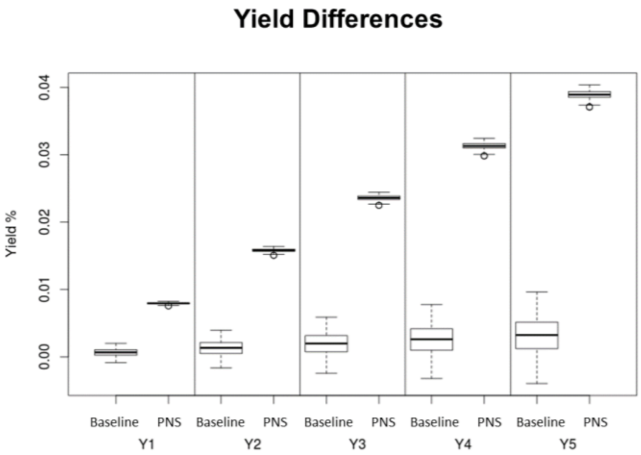


Fig. 5a. Signal and Decoy Photon Number Dependent Yield Differences.

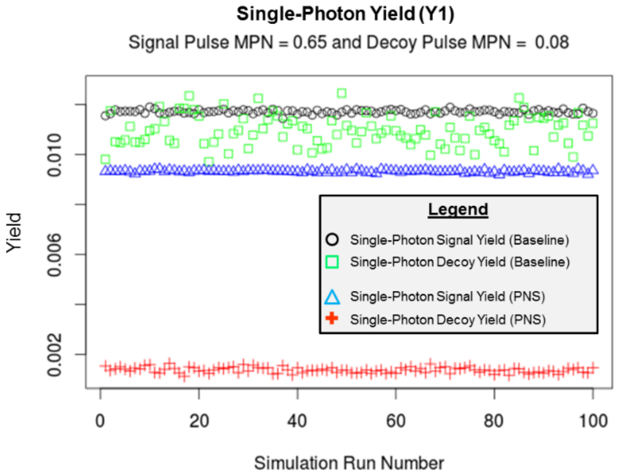


Fig. 5b. Single Photon Yield Observations (Signal and Decoy states) for a clean and dirty version.

In this figure, the green and red decoy Y1 yields appear to vary more than the black and blue signal Y1 yields. As illustrated by the Detect Eavesdropper use case, the decoy yield calculations originate from the measured number of decoy detections. This was found to be the source of the apparent variance due to the lower occurrence percentage and lower MPN (refer to Table 2) resulting in significantly fewer decoy detections than signal detections. Specifically, the mean number of decoy detections was approximately 1071 with a variance of 978 and the mean number of signal detections was 48,808 with a variance of 1081. While the actual variance in the number of detections is similar, the smaller mean number decoy detections makes the variance proportionally larger. Overall, these results provide convincing evidence to verify the decoy state enabled QKD system model was implemented correctly.

V. CONCLUSION

In this paper we examined the use of a MBSE methodology to perform requirements definition, activity decomposition, system design and analysis, model verification, and validation activities to develop a decoy state enabled QKD simulation study. We explored the use of DoDAF Viewpoints, use cases, and other tools to support development of our system-level model. We demonstrated that our decoy state enabled QKD system model was able to match experimental results, when compared to an empirical signal gain value identified in the literature. The analysis was used to develop evidence of the effectiveness of the decoy state enabled BB84 protocol to detect a PNS attack and enables further parameter studies of decoy state enabled QKD systems. This research is the first known application of MBSE to define and analyze a decoy state enabled QKD system

The results of the research highlight the value of using MBSE products, tailored for our specific stakeholder needs, to conduct analysis of complex systems and facilitate communication amongst stakeholders. Specifically we observed the following benefits:

- MBSE artifacts clarified understanding and communication of this capabilities and physical phenomena of this complex system. For example, the employment of use cases increased the team’s understanding of the decoy state protocol and helped identify dependencies between phases. These artifacts improved communication of design decisions compared the processes and methods used in our early work.

- We found that functionally decomposing and analyzing the protocol effectively guided model design and facilitated shared understanding of the new decoy state enabled BB84 protocol amongst distributed team members across multiple domains of expertise. Additionally, we observed how the functional and activity decomposition supported a seamless transition into behavior modeling and algorithm implementation.
- Furthermore, the requirements-driven approach provided traceability between early and late phases of the study which permitted incremental verification and shaped the experimental design supporting both system verification and validation. Using SE tools to capture requirements, design decisions, and identify verification methods greatly streamlined this process.

While there were significant benefits to using this approach, we also encountered a few minor issues:

- The first issue arose while decomposing the BB84 and decoy state protocols. As discussed, the OV-5a is an activity oriented model in which the activities are defined by verb phrases. This caused some concern amongst team members who were accustomed to referring to the activities as gerunds, e.g., *Sifting* (gerund) vs. *Generate Sifted Key* (verb phrase) and *Quantum Exchange* (gerund) vs. *Exchange Quantum Bit* (verb phrase).
- Additionally, we stumbled somewhat while defining subsystem actors due to the propensity of actors to be human. As noted, we settled on identifying the protocol as the actor. However, this affinity for actors to remain human is noted in the use cases where the *Alice Protocol* and *Bob Protocols* actor names are shortened to *Alice* and *Bob*.
- Furthermore, we discovered that balancing diagram complexity and clear communication of ideas was an art. For example, system level use case diagrams contain much information but are difficult to read. We found that breaking this diagram into smaller pieces which focused on a particular area was beneficial. The additional use case diagrams can be found in Appendix B.

Despite these minor issues, the benefits of the MBSE approach paid great dividends in our research.

Future research may include accomplishing sensitivity analysis by manipulating control variables as well as adding variability to the decoy state enabled QKD system model to further define the security-performance trade space.

VI. DISCLAIMER

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the US Air Force, the DoD, or the US Government.

VII. ACKNOWLEDGEMENTS

This work was supported by the Laboratory for Telecommunication Sciences (grant number 5713400-301-6448).

References

[1] L. O. Mailloux, M. R. Grimaila, D. D. Hodson, G. Baumgartner and C. McLaughlin, "Performance evaluations of quantum key distribution system architectures," *IEEE Security and Privacy*, vol. 13, no. 1, pp. 30-40, 2015.

[2] A. L. Ramos, J. V. Ferreira and J. Barcelo, "Model-based system engineering: an emerging approach for modern systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 42, no. 1, pp. 101-111, 2012.

[3] R. Cloutier, B. Sauser, M. Bone and A. Taylor, "Transitioning Systems Thinking to Model-Based Systems Engineering: Systemigrams to SysML Models," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS,* p. Accepted for publication, 2015.

[4] S. Wiesner, "Conjugate coding," *ACM Sigact News,* vol. 15, no. 1, pp. 78-88, 1983.

[5] B. Schumacher, "Quantum Coding," *Physics Review A,* vol. 51, no. 4, pp. 2738-2747, 1995.

[6] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, 1984.

[7] N. Gisin, G. Ribordy, W. Tittel and H. Zbinden, "Quantum cryptography," *Reviews of Modern Physics,* vol. 74, no. 1, pp. 145-195, 2002.

[8] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus and M. Peev, "The security of practical quantum key distribution," *Reviews of Modern Physics,* vol. 81, no. 3, pp. 1301-1350, 2009.

[9] G. S. Vernam, "Cipher printing telegraph systems for secret wire and radio telegraphic communications," *American Institute of Electrical Engineers, Transactions of the,* vol. 45, pp. 295-301, 1926.

[10] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell system technical journal,* vol. 28, no. 4, pp. 656-715, 1949.

[11] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature,* vol. 299, no. 5886, pp. 802-803, 1982.

[12] R. Renner, N. Gisin and B. Kraus, "An information-theoretic security proof for QKD protocols," *arXiv:quant-ph/0502064v1,* 2005.

[13] D. Gottesman, H. K. Lo, N. Lütkenhaus and J. Preskill, "Security of quantum key distribution with imperfect devices," *Quantum Information & Computation,* vol. 4, no. 5, pp. 325-360, 2004.

[14] V. Scarani and C. Kurtsiefer, "The black paper of quantum cryptography: real implementation problems," *arXiv:0906.4547v2,* 2009.

[15] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail and J. Smolin, "Experimental quantum cryptography," *Journal of Cryptology,* vol. 5, no. 1, pp. 3-28, 1992.

[16] B. Huttner, N. Imoto, N. Gisin and T. Mor, "Quantum cryptography with coherent states," *Physical Review A,* vol. 51, no. 3, pp. 1863-1869, 1995.

[17] G. Brassard, N. Lutkenhaus, T. Mor and B. C. Sanders, "Limitations on Practical Quantum Cryptography," *Physical Review Letters,* vol. 85, no. 6, pp. 1330-1333, 2000.

[18] N. Lutkenhaus, "Security against individual attacks for realistic quantum key distribution," *Physical Review A,* vol. 61, no. 052304, pp. 052304-1-052304-10, 2000.

[19] W.-Y. Hwang, "Quantum key distribution with high loss: Toward global secure communication," *Physical Review Letters,* vol. 91, no. 5, p. 057901, 2003.

[20] H.-K. Lo, X. Ma and K. Chen, "Decoy state quantum key distribution," *Physical Review Letters,* vol. 94, no. 3, p. 230504, 2005.

[21] X. Ma, B. Qi, Y. Zhao and H.-K. Lo, "Practical decoy state for quantum key distribution," *Physical Review,* vol. 72, no. 1, p. 012326, 2005.

[22] T. Y. Chen, H. Liang, Y. Liu, W. Q. Cai, L. Ju, W. Y. Liu and J. W. Pan, "Field test of a practical secure communication network with decoy-state quantum cryptography.," *Optics express,* vol. 17, no. 8, pp. 6540-6549, 2009.

[23] International Council on Systems Engineering, "Systems Engineering Vision 2025," 23 June 2014. [Online]. Available: http://www.incose.org/newsevents/announcements/docs/INCOSE_SE_Vision_2025.pdf. [Accessed 20 February 2015].

[24] D. Long and Z. Scott, A Primer for Model-Based Systems Engineering, Vitech Corporation, 2011.

[25] J. A. Estefan, "INCOSE Survey of MBSE Methodologies," INCOSE, Seattle, 2008.

[26] J. Holt, S. Perry, J. Bryans, Hallerstede and F. O. Handsen, "A Model-Based Approach for Requirements Engineering for Systems of Systems," *IEEE Systems Journal,* Accepted for publication.

[27] S. C. Spangelo, D. Kaslow, C. Delp, B. Cole, L. Anderson, E. Fosse, B. S. Gilbert, L. Hartman, T. Kahn and J. Cutler, "Applying model based systems engineering (mbse) to a standard cubesat," *In Aerospace Conference, 2012 IEEE,* pp. 1-20, 2012.

[28] A. Soyler and S. Sala-Diakanda, "A Model-Based Systems Engineering Approach to Capturing Disaster Management Systems," *In Systems Conference, 2010 4th Annual IEEE,* pp. 283-287, 2010.

[29] J. Banks, J. S. Carson II, B. L. Nelson and D. M. Nicol, Discrete Event System Simulation, 5, Ed., Upper Saddle River: Prentice Hall, 2010.

[30] DoD Deputy Chief Information Officer, "DoDAF Viewpoints and Models," 2010. [Online]. Available: http://dodcio.defense.gov/TodayinCIO/DoDArchitectureFramework/dodaf20_viewpoints.aspx.

[31] H. Gomaa, Software Modeling & Design, Cambridge: Cambridge University Press, 2011.

[32] S. W. Ambler, "Web services programming tips and tricks: Documenting a use case," IBM, 05 October 2000. [Online]. Available: http://www.ibm.com/developerworks/library/ws-tip-docusecase/. [Accessed 14 01 2015].

[33] G. Schneider and J. P. Winters, "Chapter 7: Documenting Use Cases," [Online]. Available: http://www.ibm.com/developerworks/rational/library/content/legacy/parttwo/1000/0670/0670_Schneider_Ch07.pdf. [Accessed 14 01 2015].

[34] L. O. Mailloux, J. D. Morris, M. R. Grimaila, D. D. Hodson, D. R. Jacques, J. M. Colombi, C. McLaughlin, R. Engle and J. Holes, "A modeling framework for studying quantum key dstribution system implementation non-idealities," *IEEE Access,* Accepted for Publication.

[35] C/S2ESC - Software & Systems Engineering Standards Committee, "1362-1998 - IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document," IEEE Computer Society, 1998.

[36] Defense Acquisition University, "Concept of Operations," 2015. [Online]. Available: https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=f8f70a76-5dda-4346-9bfb-7ef0608f71bb.

[37] MITRE, "Concept of Operations," 2015. [Online]. Available: http://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/concept-development/concept-of-operations.

[38] SE Handbook Working Group International Council on Systems Engineering, "Systems Engineering Handbook," INCOSE, San Diego,

2011.

[39] DoD Deputy Chief Information Officer, "Operational Viewpoint (OV-1)," U.S. Department of Defense, [Online]. Available: http://dodcio.defense.gov/TodayinCIO/DoDArchitectureFramework/dodaf20_ov1.aspx. [Accessed 13 01 2015].

[40] J. O. Grady, System Verification, Burlington: Academic Press, 2007.

[41] DoD Deputy Chief Information Officer, "Operational Viewpoint (OV-5a & OV-5b)," 2010. [Online]. Available: http://dodcio.defense.gov/TodayinCIO/DoDArchitectureFramework/dodaf20_ov5ab.aspx. [Accessed 13 01 2015].

[42] S. Friedenthal, A. Moore and R. Steiner, A Practical Guide to SysML, Waltham: Elsevier, 2012.

[43] L. O. Mailloux, R. D. Engle, M. R. Grimaila, D. D. Hodson and C. McLaughlin, "Modeling decoy state quantum key distribution systems," *Journal of Defense Model and Simulation,* Submitted 2014.

[44] ISO/IEC, "Systems and software engineering — System life cycle," International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva, 2008.

[45] Defense Acquisition University, "[Systems Engineering] Introduction," Defense Acquisition University, 2014. [Online]. Available: https://acc.dau.mil/CommunityBrowser.aspx?id=638297. [Accessed 03 12 2014].

[46] A. M. Law and D. W. Kelton, Simluation Modeling and Analysis, Third ed., Boston: McGraw Hil, 2000.

[47] R. G. Sargent, "Verification and Validation of Simluation Models," in *2010 Winter Simulation Conference*, Baltimore, 2010.

[48] O. Balci, "Verification, Validation, and Certification of Modeling and Simulation Applications," in *2003 Winter Simulation Conference*, New Orleans, 2003.

[49] J. Banks, J. S. Carson II, B. L. Nelson and D. M. Nicol, "BCNN Homepage," 2010. [Online]. Available: http://www.bcnn.org/. [Accessed 01 2015].

VI. Additional Results and Analyses

Overview

The purpose of this chapter is to capture all research products and analyses not fully elaborated in the journal papers presented in Chapters IV and V. Specifically, this chapter provides insight into the modeling development process, introduces DoDAF viewpoints and use cases created during the research. Lastly, this chapter presents additional significant implementation findings and experimental results related to experimentation.

Model Development Process

Figure 15 shows a portion of the model development process presented in Figure 14. In this research, steps 1 (*Problem formulation)* and 2 (*Setting of objectives and overall project plan)*, were accomplished by the identification of the research goal, objective, and questions in Chapter I and the methodology described in Chapter III. The initial conceptual model, model translation and data collection steps 3-5, were detailed in the modeling decoy state protocol journal article in Chapter IV. Model translation is described in the following sections which briefly discuss two approaches to model translation.

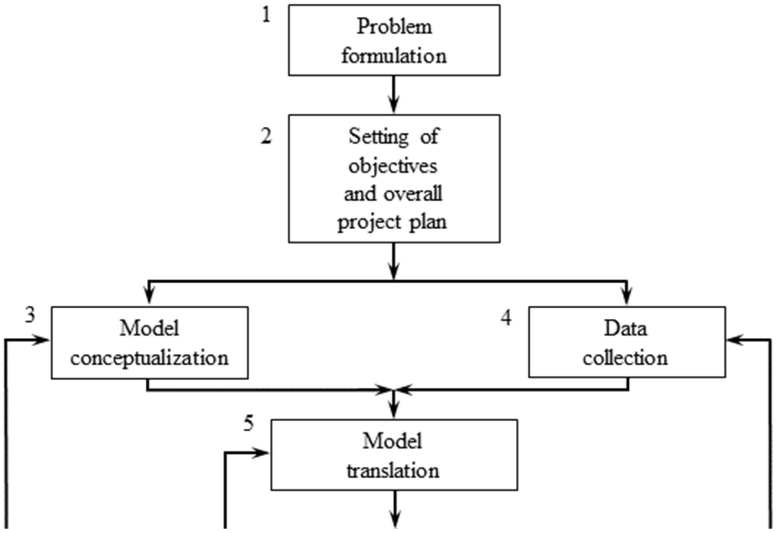


Figure 15. Partial Reproduction of Model Development Process (Chapter III, Figure 14).

Approach #1 – "From Scratch"

Initially, building a system model "from scratch" was considered as a method of fully exercising the Model Development Process. Conceptually, the plan was to use existing optical component modules as well as create new controllers to direct the protocol. In this design, Alice would employ existing components including a laser, an attenuation mechanism, and controllers (to include timing regulation), while Bob would be constructed from existing detectors and require implementation of new controlling and timing components. Alice and Bob could be connected using existing quantum and public channels.

Through discussion with the sponsor and team members, the author discovered this approach would not take advantage of the existing reference model architecture. In a sense, the author would need to "reinvent the wheel" in order to recreate the control logic already implemented in the existing model. Additionally, this approach was arguably inconsistent with the sponsor's desire to have a flexible and reusable system model. In a sense, the sponsor's system requirement, i.e. reuse, would not have been met. Furthermore, this approach would effectively ignore the lessons learned which resulted from the initial design. Neither of these effects is consistent with a sound MBSE approach.

The resulting conclusion was to develop a system model that reuses existing optical and control components and only make modifications where required i.e., the capabilities of the decoy state enabled QKD system (or sponsor) required it. *Approach #2* outlines the design decisions made to shape the final model.

Approach #2 – "Maximum Reuse"

After determining that building the model from scratch would be less efficient, the author adopted a modeling approach promoting the maximum reuse of existing optical, timing, and controlling components. The existing work included a toolkit of hardware component modules and a notional architecture that was designed to explore the behavior of non-ideal QKD systems. In this notional architecture, the BB84 protocol was implemented and work had begun to implement decoy states. Additionally, the framework components had been verified and the notional architecture itself had been validated in support of other research efforts. More specifically, the notional architecture was able to randomly select, prepare and measure signal, decoy and vacuum state pulses [96, 81, 95]. Thus, the research could focus on defining, designing and implementing the decoy state protocol without starting from scratch.

DoDAF Viewpoint Artifacts

In addition to the OV-1, OV-5a, and SV-1 artifacts discussed in Chapter V, the DoDAF Operational Activity Model (OV-5b) facilitated understanding of the "As-Is" state of the notional architecture. The OV-5a and OV-5b Viewpoints are designed to decompose activities that are necessary to perform mission level activities [97]; in this study, the author tailored the use of these products and defined the mission as the execution of the BB84 protocol. In order to define the "As-Is" state, OV-5b level 0 and OV-5b level 1 diagrams were created to describe the BB84 protocol and were based on previous work by the research team [98].

The two levels of OV-5b were created to capture both high and low levels of detail in the activity model. The rationale behind creating the OV-5b Level 0 and OV-5b Level 1 was to experiment with the amount of information that could be effectively communicated in each level as described below. Additionally, levels of detail are consistent with the tailorable philosophy of employing MBSE and SysML [33, 54].

The OV-5b level 0 is provided as Figure 16. The eight phases identified in the OV-5a are now linked together to show the progression of activities or the activity flow. Activity flows are indicated by the dashed lines which terminate with an arrow on one side. The arrow indicates which activity will be completed next. The labels on these activity flows indicate what information is passed between activities. Finally, the diamonds indicate that a decision point has been reached.

The activities in Figure 16 contain both the BB84 verb phrase, which conforms to UML guidance, and the gerund-based phrases typically associated with QKD systems. This was an effort made by the researcher to provide clarity to the research team stakeholders rather than strictly adhere to UML standards, which is consistent with the tailorable nature of MBSE [33].

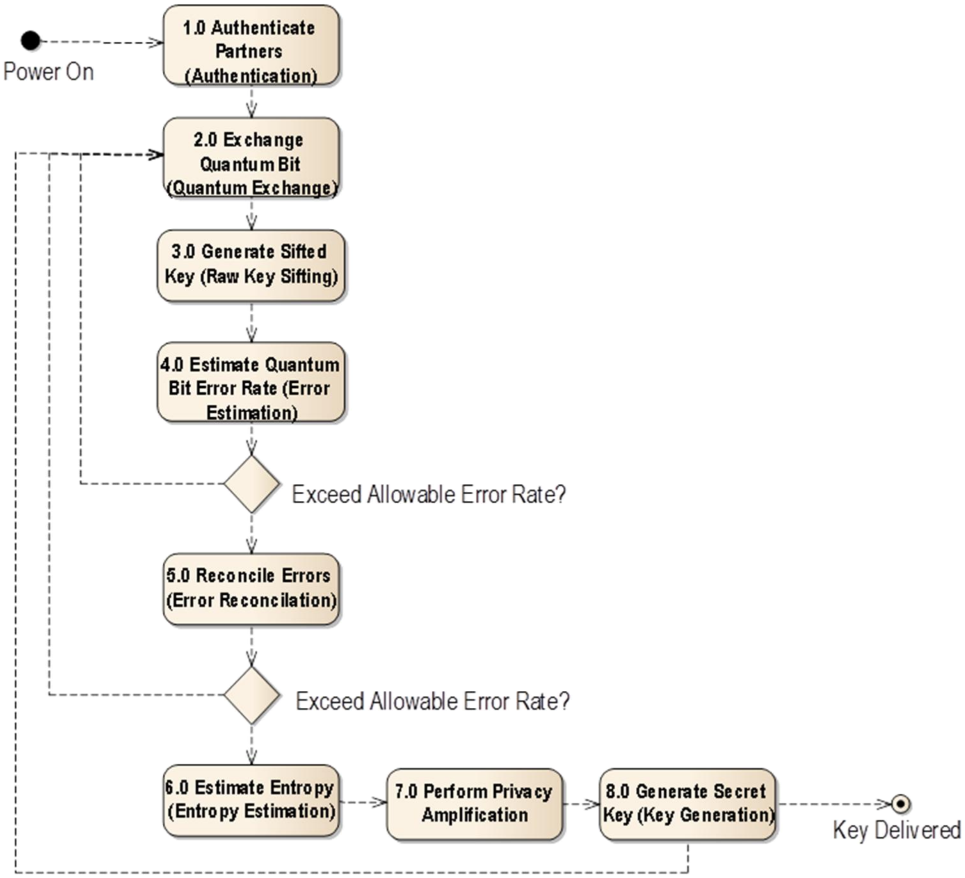

Power On
1.0 Authenticate Partners (Authentication)
2.0 Exchange Quantum Bit (Quantum Exchange)
3.0 Generate Sifted Key (Raw Key Sifting)
4.0 Estimate Quantum Bit Error Rate (Error Estimation)
Exceed Allowable Error Rate?
5.0 Reconcile Errors (Error Reconcilation)
Exceed Allowable Error Rate?
6.0 Estimate Entropy (Entropy Estimation)
7.0 Perform Privacy Amplification
8.0 Generate Secret Key (Key Generation)
Key Delivered

Figure 16. BB84 Operational Activity Model - OV-5b Level 0.

The OV-5b level 1 contains substantially more detail than the OV-5b Level 0 and provides designated areas of responsibility through the use of “swim lanes.” Swim lanes are shown in this series of OV-5b diagrams to indicate which activities are the responsibilities of Alice or Bob. Alice's swim lane is colored in blue and Bob's swim lane is orange.

Activities that require action from both Alice and Bob are placed on the line which separates Alice and Bob's swim lanes. The jagged or zig-zag nature of the activity flows crossing between Alice's and Bob's swim lanes are intended to highlight that flow involves communication between both Alice and Bob. Figure 17 and Figure 18 depict the OV-5b broken into two parts.

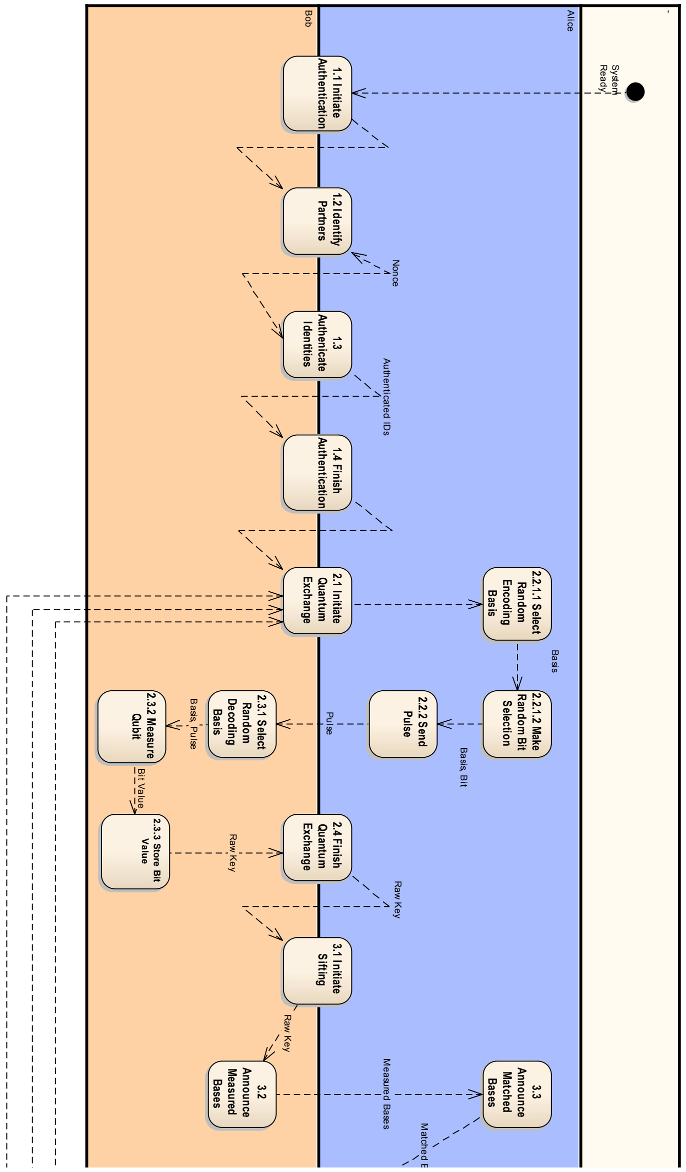

Alice
Bob
System Ready
1.1 Initiate Authentication
1.2 Identify Partners
Nonce
1.3 Authenicate Identities
Authenticated IDs
1.4 Finish Authentication
2.1 Initiate Quantum Exchange
2.2.1.1 Select Random Encoding Basis
Basis
2.2.1.2 Make Random Bit Selection
Basis, Bit
2.2.2 Send Pulse
Pulse
2.3.1 Select Random Decoding Basis
Basis, Pulse
2.3.2 Measure Qubit
Bit Value
2.3.3 Store Bit Value
Raw Key
2.4 Finish Quantum Exchange
Raw Key
3.1 Initiate Sifting
Raw Key
3.2 Announce Measured Bases
Measured Bases
3.3 Announce Matched Bases
Matched B

Figure 17.BB84 Operational Activity Model OV-5b Level 1 (1/2).

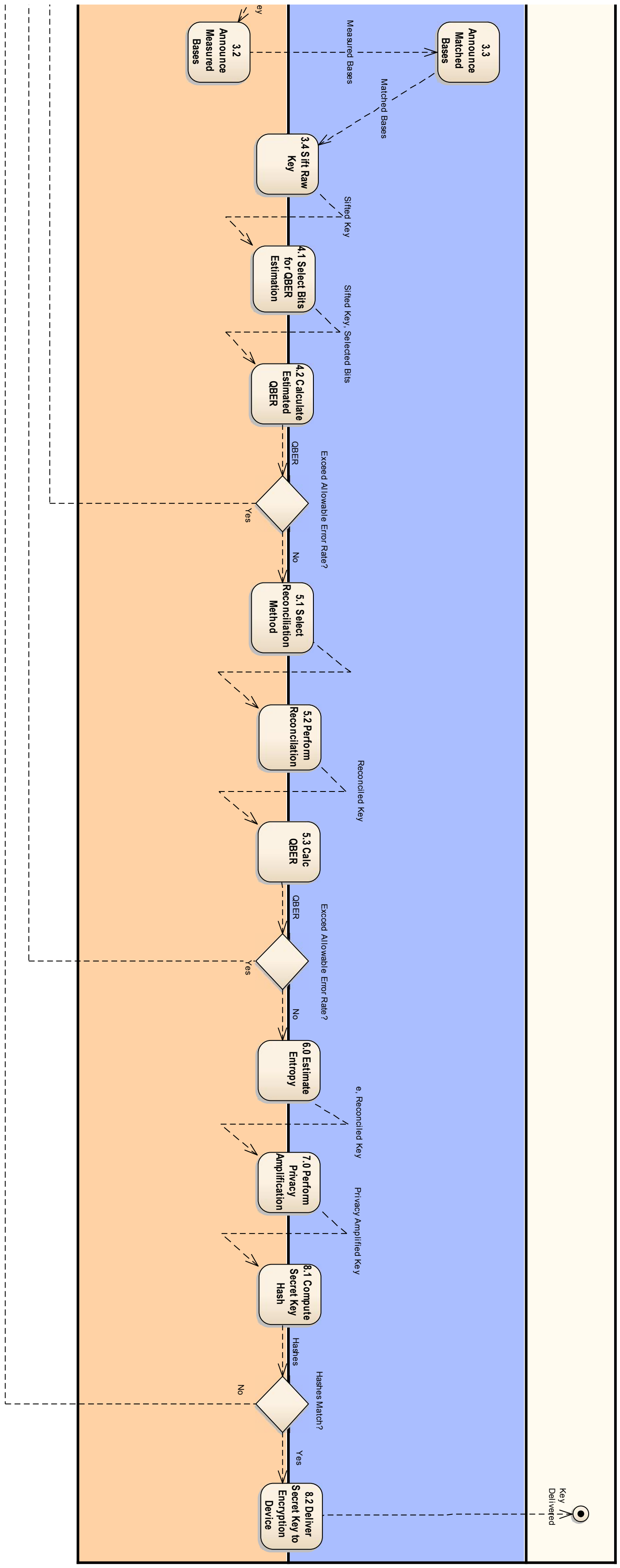

ey
3.2 Announce Measured Bases
Measured Bases
3.3 Announce Matched Bases
Matched Bases
3.4 Sift Raw Key
Sifted Key
4.1 Select Bits for QBER Estimation
Sifted Key, Selected Bits
4.2 Calculate Estimated QBER
QBER
Exceed Allowable Error Rate?
Yes
No
5.1 Select Reconciliation Method
5.2 Perform Reconciliation
Reconciled Key
5.3 Calc QBER
QBER
Exceed Allowable Error Rate?
Yes
No
6.0 Estimate Entropy
e, Reconciled Key
7.0 Perform Privacy Amplification
Privacy Amplified Key
8.1 Compute Secret Key Hash
Hashes
Hashes Match?
No
Yes
8.2 Deliver Secret Key to Encryption Device
Key Delivered

Figure 18. BB84 Operational Activity Model OV-5b Level 1 (2/2).

After the creation of the OV-5a pair of diagrams, the use case development work began. Still, the “As-Is” OV-5b were painstakingly created to support a presentation at the INCOSE Great Lakes Regional Conference in October 2014 [99]. However these products were difficult to visualize outside of the tool in which they were created. Additionally, the use cases were found to capture and communicate more information, e.g., captured algorithms and equations, than the OV-5b’s and provided a more readable format. For these reasons, “To-Be” versions of the OV-5b Operational Activity Model were not created.

Use Cases Development

The research employed the use case approach to capture additional details of the “As-Is” activities in the notional architecture as well as clarify capability gaps, and refine essential details of the desired decoy state enabled QKD system. Detailing the “As-Is” architecture enabled a smooth transition to design and implement the “To-Be” model. The use case modeling effort solidified and combined the knowledge gained from the identification of operational activities as well as knowledge gained from reviewing the existing code from the notional architecture.

Table 3 shows the use case template used to define the studied use cases. As noted in Chapter II, this template was a combination of Gomaa's and RUP-based templates [12, 66, 67]. Additionally, the research sponsor requested that information known by Alice and Bob should be capture in the pre-conditions and post-conditions.

Table 3. Use Case Template [12, 66, 67].

Use case name: Clearly identify the purpose of the use case and typically begins with a verb.

Summary: Terse description of the use case.

Dependency: Identify <<Includes>> or <<Extends>> relationships of the use case.

Actor(s): Identify the actors associated with the use case.

Resources: Identify applicable resources required by the use case.

Pre-condition: Explain or identify the conditions that must be met before the use can begin.

Post-condition: Explain or identify the conditions that will normally exist at the end of the use case.

Main sequence: Typical sequence of events that define the use case.

Alternative sequence: Alternative or exception events that deviate from the main sequence.

Design Decisions: Capture relevant decisions that affected the use case design.

Outstanding questions/issues: Identify or explain issues that remain after the design/definition of the use case.

Figure 19 is the complete use case diagram depicting the all the relevant actors, system and subsystem boundaries, and all use cases. The red rectangle denotes the QKD system boundary whereas the purple rectangle outlines the system of interest's boundary. Actors on the left side of the diagram are human actors and actors on the right are defined by the protocol. Use cases are identified by the bubbles. Blue designates use cases that are exclusive to the standard BB84 protocol and orange is used for use cases that are exclusive to the Decoy State Enabled BB84 protocol. Green identifies use cases which are shared by both protocols. Due to the complexity of this figure, pertinent elements are examined separately in Figures 20-22. From Figure 19, the reader should merely identify the system boundaries and actors.

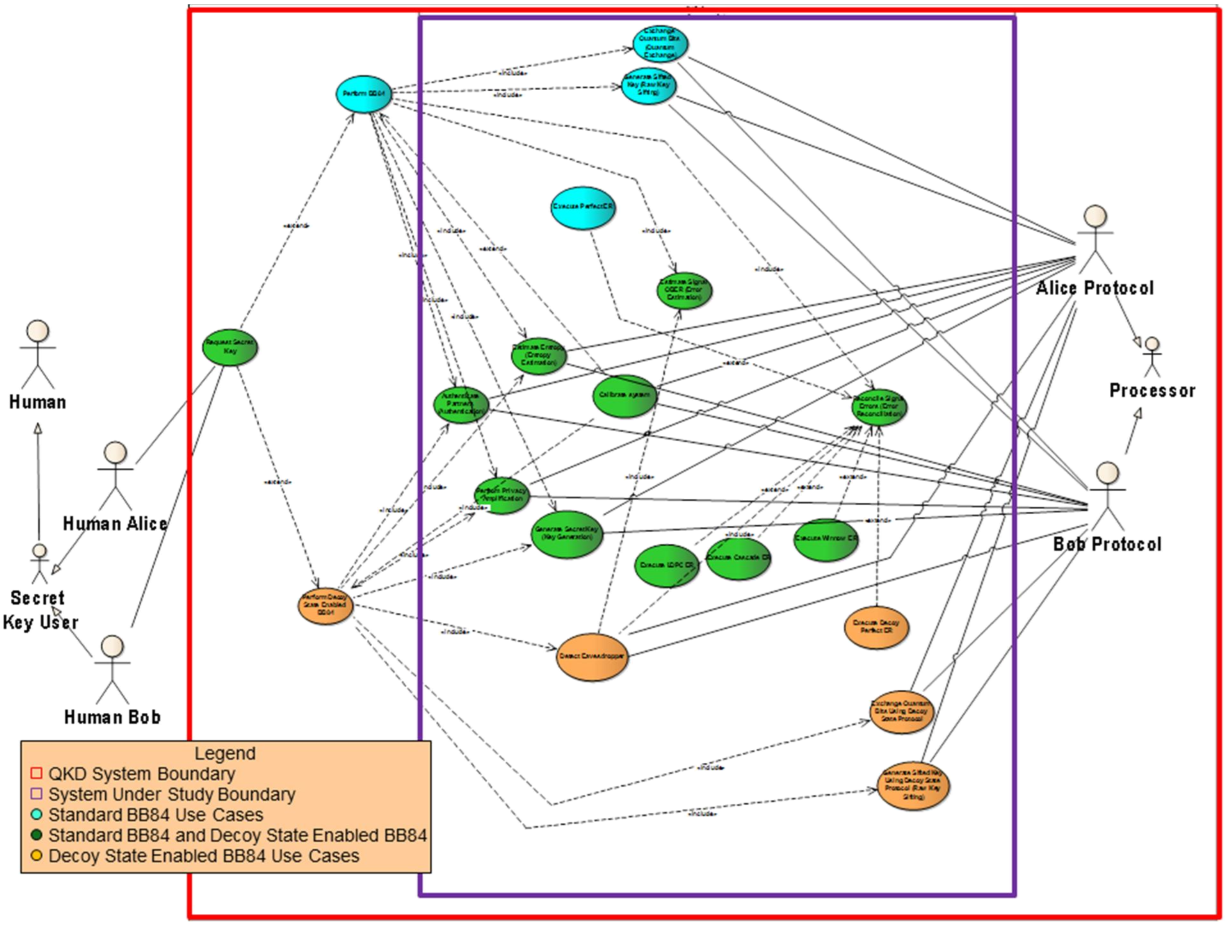

Exchange Quantum Bits (Quantum Exchange)
Generate Sifted Key (Raw Key Sifting)
Perform BB84
«include»
«extend»
Execute PerfectER
Estimate Signal QBER (Error Estimation)
Request Secret Key
Estimate Entropy (Entropy Estimation)
Authenticate Partners (Authentication)
Calibrate system
Reconcile Signal Errors (Error Reconciliation)
Perform Privacy Amplification
Generate SecretKey (Key Generation)
Execute LDPC ER
Execute Cascade ER
Execute Winnow ER
Perform Decoy State Enabled BB84
Detect Eavesdropper
Execute Decoy Perfect ER
Exchange Quantum Bits Using Decoy State Protocol
Generate Sifted Key Using Decoy State Protocol (Raw Key Sifting)
Human
Human Alice
Secret Key User
Human Bob
Alice Protocol
Processor
Bob Protocol
Legend
□ QKD System Boundary
□ System Under Study Boundary
○ Standard BB84 Use Cases
● Standard BB84 and Decoy State Enabled BB84
○ Decoy State Enabled BB84 Use Cases

Figure 19. Complete QKD Use Case Diagram.

Figure 20 displays the human actor relationship between the use cases and the actors. From this diagram, observe that both the human actor versions of Alice and Bob belong to both Secret Key Users and Human actor types. Additionally, one should observe that their only interaction with the QKD System is through the *Request Secret Key* use case which supports both the standard and decoy state enabled BB84 protocol. This is possible because the *Request Secret Key* use case is extended by the *Perform Decoy State Enabled BB84* (shown) and *Perform BB84* (not shown) use cases. The *Perform BB84* use case and the remaining standard BB84 use cases are displayed in Figure 21.

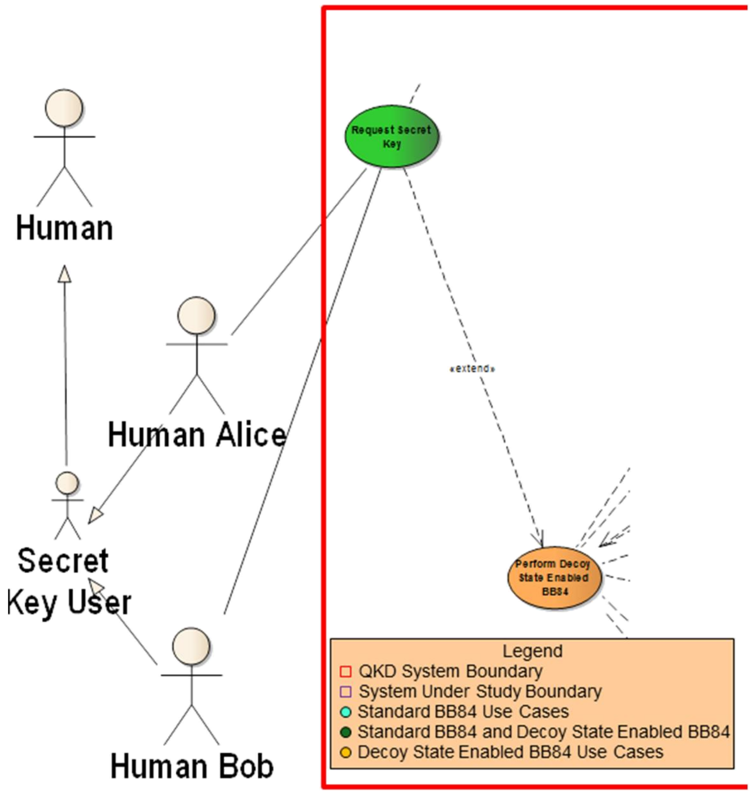

Human
Human Alice
Secret Key User
Human Bob
Request Secret Key
«extend»
Perform Decoy State Enabled BB84
Legend
□ QKD System Boundary
□ System Under Study Boundary
○ Standard BB84 Use Cases
● Standard BB84 and Decoy State Enabled BB84
○ Decoy State Enabled BB84 Use Cases

Figure 20. Left Side of QKD Use Case Diagram.

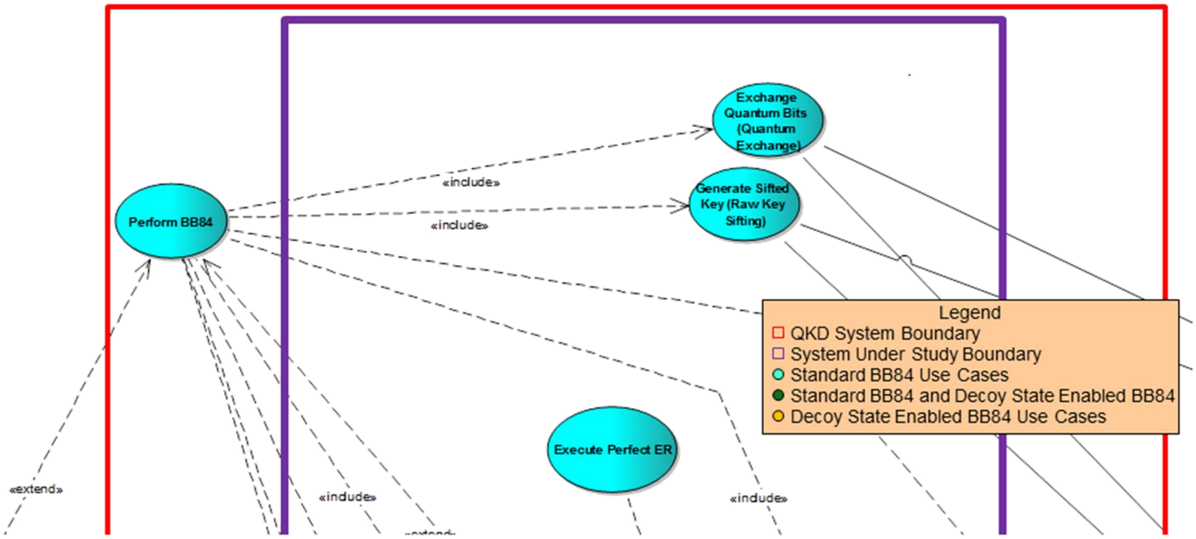

Exchange Quantum Bits (Quantum Exchange)
Generate Sifted Key (Raw Key Sifting)
Perform BB84
«include»
«include»
Execute Perfect ER
«extend»
«include»
«extend»
«include»
Legend
□ QKD System Boundary
□ System Under Study Boundary
○ Standard BB84 Use Cases
● Standard BB84 and Decoy State Enabled BB84
○ Decoy State Enabled BB84 Use Cases

Figure 21. Use Case Diagram BB84 Exclusive Use Cases.

The use case diagram shown in Figure 22 captured and focused the remaining details required to organize the use cases for the decoy state protocol. While the color scheme remains the same as in the previous diagrams, the protocol actors have switched sides and the use case bubbles have been organized to illustrate those which are necessary to execute the decoy state protocol. In this diagram, one can clearly identify the relationships between the actors and uses cases. Combined with the inset SV-1 at the top left, this diagram communicates the intended boundaries for the system of interest.

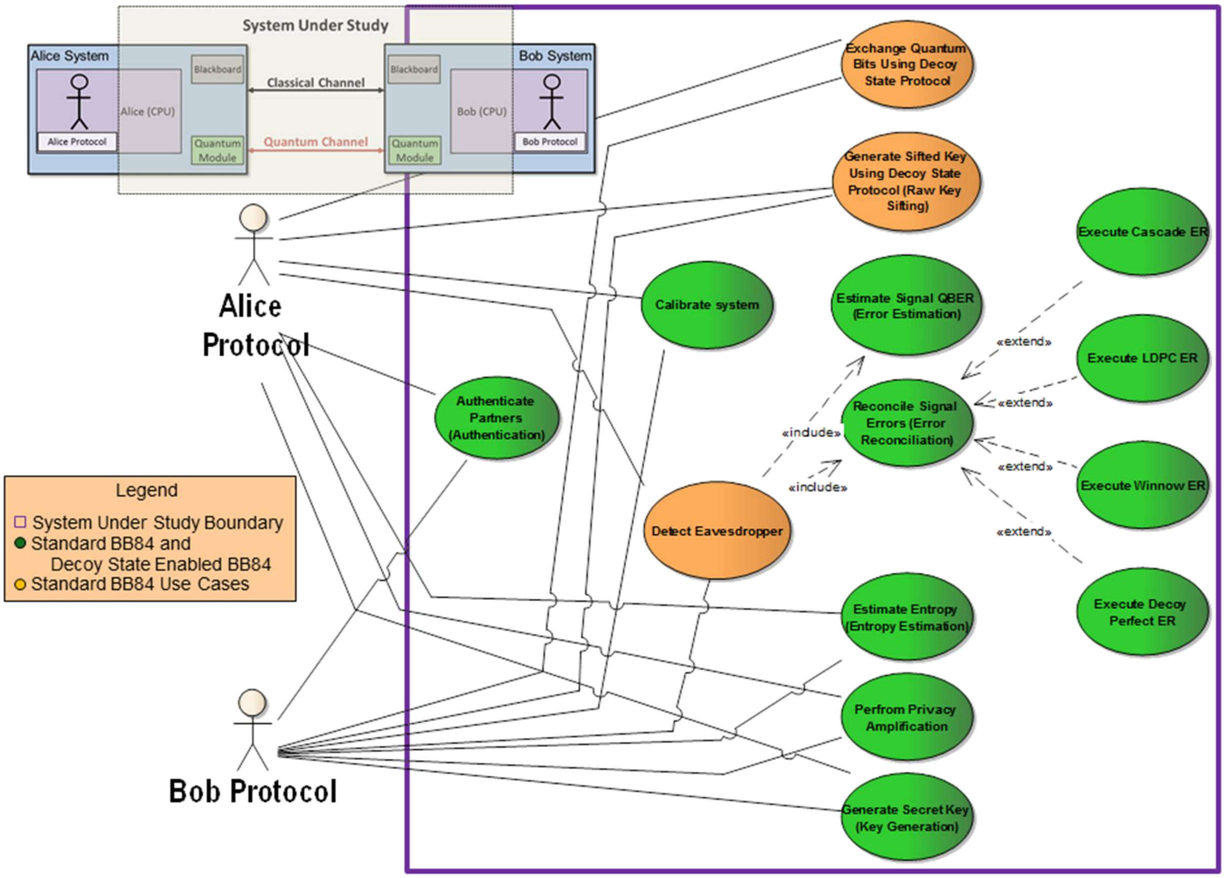

System Under Study
Alice System
Alice Protocol
Alice (CPU)
Blackboard
Quantum Module
Classical Channel
Quantum Channel
Blackboard
Quantum Module
Bob (CPU)
Bob System
Bob Protocol
Alice Protocol
Bob Protocol
Legend
□ System Under Study Boundary
● Standard BB84 and Decoy State Enabled BB84
○ Standard BB84 Use Cases
Exchange Quantum Bits Using Decoy State Protocol
Generate Sifted Key Using Decoy State Protocol (Raw Key Sifting)
Calibrate system
Estimate Signal QBER (Error Estimation)
Authenticate Partners (Authentication)
Reconcile Signal Errors (Error Reconciliation)
Detect Eavesdropper
«include»
«include»
«extend»
«extend»
«extend»
«extend»
Execute Cascade ER
Execute LDPC ER
Execute Winnow ER
Execute Decoy Perfect ER
Estimate Entropy (Entropy Estimation)
Perfrom Privacy Amplification
Generate Secret Key (Key Generation)

Figure 22. Use Case Diagram with SV-1 - Decoy State Enabled BB84.

From the complete set of DoDAF Viewpoints and use case artifacts created, a set of behaviors, calculations, formulas, and measures of performance, and requirements were defined, analyzed, and thoroughly documented. The fully dressed use cases, which include the calculations and formulas, are included in Appendix C. Decoy State Enabled QKD System Use Cases and Appendix D. Other BB84 Use Cases. The refined requirements are included in Appendix F: Requirements. This set of products was sufficient for the researcher to begin implementing the decoy state protocol which began a new iteration of *Model translation* step in the Model Development Process.

Implementation Findings

This section discusses noteworthy findings and lessons learned during model implementation. The first is related to design decisions regarding the protocol and the second pertains to complexities involving timing synchronization in the QKD system model.

Protocol Design Decisions

Configuring the model to meet the protocol specifications involved implementing the steps outlined in the use cases found in Appendix C. Decoy State Enabled QKD System Use Cases. The gap between the conceptual protocol definition and the desired model provided many opportunities for design and implementation decisions. Examples include tradeoff between software design patterns, network traffic efficiency, and others simply involved strengthening previous decisions to make Alice fulfill a leader role and Bob fulfill the role of the follower. Typically these design tradeoffs were captured in the previously mentioned use cases, while implementation decisions were captured in software documentation.

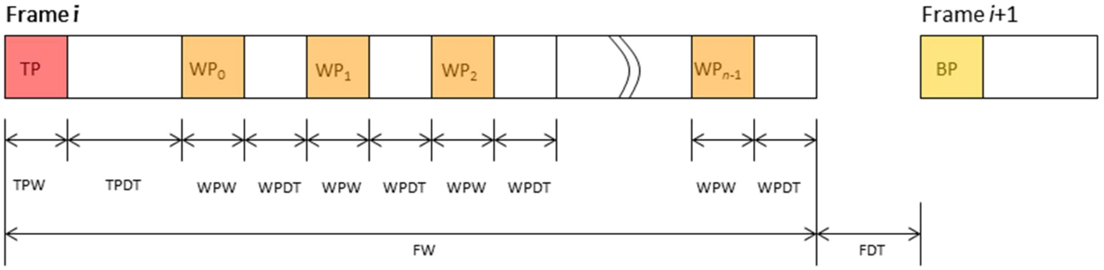
A noteworthy example of the tradeoff between competing software engineering design principles occurred during the implementation of the sifting phase. Specifically, inheritance and polymorphism were considered against simply modifying the existing code for the sifting process. Whereas the former option provides potential for future code reuse, it would require additional redesign of the working sifting process. As there was no planned future requirement to reuse this section of the code, a simple direct modification of the pertinent sections was the consensus answer from the team.

Thus the sections of code involved in sifting were changed to conform to the sequence described in the use case.

Timing Synchronization

One of the most critical and challenging aspects of designing and implementing the system model involved timing synchronization. Moreover, Alice and Bob exhibit both synchronous and asynchronous behaviors during the quantum exchange phase. During the measurement portion of the quantum exchange phase, Bob relies on both accurate and precise timing to detect weak coherent pulses and thus correctly measure encoded bit values. Specifically, Bob must know when to expect a pulse (accuracy) and his detectors must be ready, i.e., gated, within 10s of picoseconds (precision) prior to the pulse's arrival. Furthermore, gating or the gating period is the time period during which Bob's SPDs are ready and able to detect a weak coherent pulse. Gating is performed to reduce noise (and therefore errors) at Bob's SPDs [81]. As a complementary pair of participants, Alice must ensure to carefully prepare and order the timing of weak coherent pulses in order for Bob make correct measurements.

This ordering of pulses is facilitated by defining an optical pulse frame to which Alice and Bob agree to adhere as shown in Figure 23, which illustrates how the optical pulse frame was conceptualized for implementation in the model. Using this approach, the number of weak coherent pulses in a frame, the duration of the pulses, the down time between pulses, and the time between frames must be defined. The use of a timing (or bright) pulse at the beginning of each frame allows Bob to identify and gate his SPDs for the arrival of trailing WCPs. Thus Bob can schedule all of the gating periods for a particular frame by knowing this information. Moreover, by breaking up the quantum exchange phase into optical pulse frames, Alice and Bob are able to ensure synchronization for the duration of the quantum exchange. Therefore, Alice and Bob have a practical method for controlling (and securing) the timing synchronization which is used for preparing and measuring weak coherent pulses.



i: Frame Identifier
n: Number of Weak Coherent Pulses in a Frame
TPW: Timing Pulse Width
TPDT: Timing Pulse to Weak Coherent Pulse Down Time

WP: Weak Coherent Pulse
WPW: Weak Coherent Pulse Width
WPDT: Weak Coherent Pulse Down Time
FW: Frame Width
FDT: Frame Down Time

Figure 23. Pulse Frame Definition.

In the original implementation, all of the timing information was hardcoded into the model and thus inflexible in terms of an easily reconfigurable model or exploiting the capabilities of OMNeT++ to configure simulation parameters. The author seized the opportunity to increase the flexibility of the model, by converting these timing values into configurable parameters via OMNeT++ NED files. While seemingly simplistic, this change required weeks of work to understand 100s of lines of code and perform reverse engineering of the antiquated timing structure. This change further enables a flexible subsystem implementation capability. Specifically this means that configuration of the compound modules, such as the quantum module in Alice or Bob, can be configured using parameters that do not need to be fixed prior to the simulation's compile time. Thus, multiple experiments and trials can be configured and executed with relative ease. This change also enables the research team to further study timing related issues such as SPD recovery times, gating periods, and fast gating [81, 100].

In order to maintain a flexible and realistic design, Alice and Bob must adhere to the optical pulse frame and account for their internal optical path timing delays. They should not be required to know any information about the other's timing delay. However, the original implementation required Bob to have knowledge about the timing delays that were internal to Alice in order to correctly detect weak coherent pulses and measure bit values. Specifically, Bob had to know the difference between the arrival times of a timing pulse and a weak coherent pulse at Alice's Output Power Monitor. This condition is inconsistent with the previously mentioned goal and is an unrealistic expectation for practical QKD systems. In terms of system realism, it is more practical for Alice and Bob to conform to a standard rather than to rely on having intimate knowledge of each other's internal delays or configuration.

To illustrate the issue, Figure 24 depicts the path that optical pulses follow inside of Alice's Quantum Module. The top half of the figure contains the quantum module, its compound modules and the connections between them. Fiber optical paths are red and electrical paths are black. The Classical Pulse Generator (CPG), the Pulse Modulator (PM), the Decoy State Generator (DSG), the Optical Security Layer (OSL), the Timing Pulse Generator (TPG), and the Output Power Monitor (OPM) are all modules through which pulses propagate. The lower half of the figure contains exploded views of the internal components and connections in the CPG (left) and the TPG (right). All optical connections are bi-directional, yet, in general, optical pulses travel from devices on the left through devices on the right. Pulses traveling from the OSL to the TPG only pass through the TPG's wave division multiplexor (WDM).

Pulses generated by the laser in the CPG propagate through multiple optical components, each with a unique propagation delay before reaching the OPM. Timing Pulses, on the other hand, are generated by the TPG's laser and follow a considerably shorter path before reaching the OPM. In order for the pulses to conform to the pulse frame specifications, Alice must know the time difference of arrival of the timing pulse and a weak coherent pulse at the TPG's Wave Division Multiplexor (WDM). That is, Alice must know the minimum amount of time that will pass from the timing pulse's and a weak coherent pulse's arrival at the OPM. This difference is proportional to the propagation delays of each optical component (including fiber) connect between the laser in the CPG and the TPG's WDM.

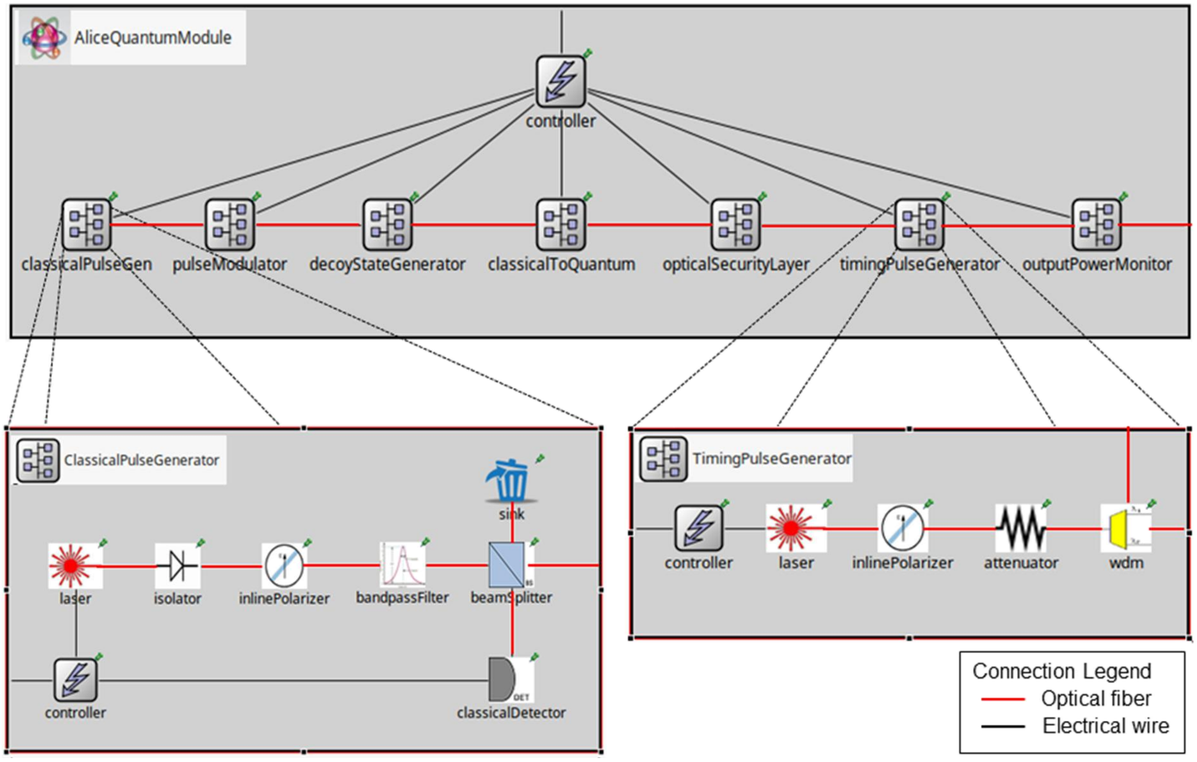


Figure 24. Alice's Quantum Module and Connections.

Similarly, Figure 25 depicts the optical path inside of Bob’s Quantum Module. The left side of the figure contains the quantum module, its modules and the connections between them where optical connections are designated in red and electrical connections in black. Optical pulses enter the quantum module and pass through the Input Stage and the Polarization Adjustment module, before being split by another WDM to the Polarization Detector (PD) or the Timing Analyzer (TA). Timing pulses are identified by wavelength and are routed to the TA’s classical detector. The classical detector sends a message to the timing analyzer controller which then schedules gating periods. Weak coherent pulses are routed to the polarization detector which performs passive basis selection and send the pulse to the appropriate SPD which enables Bob to properly decode and measure the encoded bit values. The right side of the figure contains an exploded view of the PD (top right) and TA (bottom right).

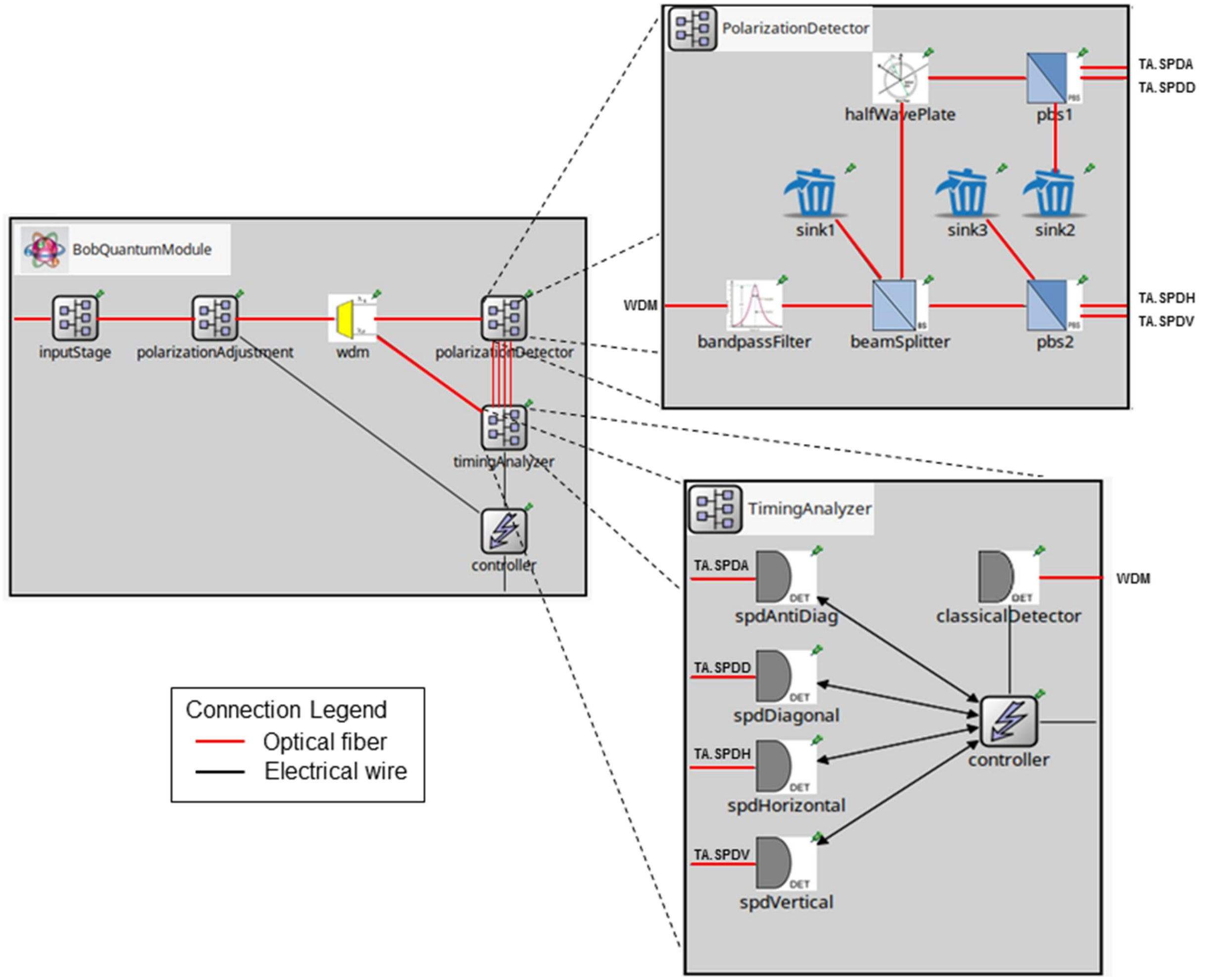
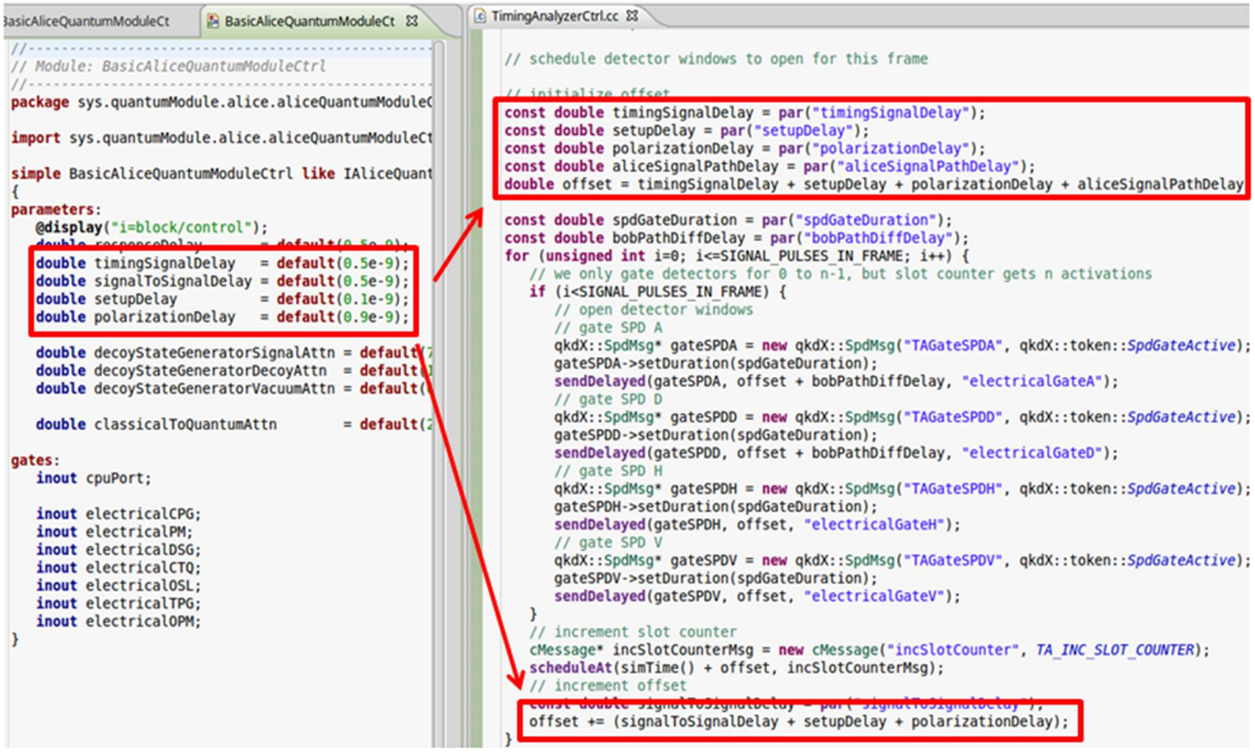


Figure 25. Bob's Quantum Module and Connections.

In order to conform to the optical frame specifications, Bob must know the time difference of arrival between a pulse at the TA’s classical detector (CD) and a pulse at the spdHorizontal (or spdVertical). Additionally, Bob must also know the time difference of arrival between a pulse at the spdHorizontal (or spdVertical) and a pulse at the spdAntiDiag (or spdDiagonal). As discussed concerning the initial implementation, Bob was also required to have knowledge of the timing difference in Alice’s Quantum Module.

Figure 26 illustrates the undesirable coupling of these timing parameters between Alice and Bob. Specifically it shows how timing parameters from Alice’s Quantum Module (left) were used in the

Bob’s Timing Analyzer (right). The parameters of interest are circumscribed by a red box, where the left side of the figure shows Alice’s timing parameters defined in her Quantum Module NED file. The arrows pointing to the right side of the figure show where these parameters are used in the Bob’s implementation of the Timing Analyzer.



```
//----------------------------------------------------
// Module: BasicAliceQuantumModuleCtrl
//----------------------------------------------------
package sys.quantumModule.alice.aliceQuantumModuleC

import sys.quantumModule.alice.aliceQuantumModuleCt

simple BasicAliceQuantumModuleCtrl like IAliceQuant
{
parameters:
   @display("i=block/control");
   double responseDelay       = default(0.5e-9);
   double timingSignalDelay   = default(0.5e-9);
   double signalToSignalDelay = default(0.5e-9);
   double setupDelay          = default(0.1e-9);
   double polarizationDelay   = default(0.9e-9);

   double decoyStateGeneratorSignalAttn = default(7
   double decoyStateGeneratorDecoyAttn  = default(1
   double decoyStateGeneratorVacuumAttn = default(

   double classicalToQuantumAttn        = default(2

gates:
   inout cpuPort;

   inout electricalCPG;
   inout electricalPM;
   inout electricalDSG;
   inout electricalCTQ;
   inout electricalOSL;
   inout electricalTPG;
   inout electricalOPM;
}
```

```
// schedule detector windows to open for this frame

// initialize offset
const double timingSignalDelay = par("timingSignalDelay");
const double setupDelay = par("setupDelay");
const double polarizationDelay = par("polarizationDelay");
const double aliceSignalPathDelay = par("aliceSignalPathDelay");
double offset = timingSignalDelay + setupDelay + polarizationDelay + aliceSignalPathDelay

const double spdGateDuration = par("spdGateDuration");
const double bobPathDiffDelay = par("bobPathDiffDelay");
for (unsigned int i=0; i<=SIGNAL_PULSES_IN_FRAME; i++) {
   // we only gate detectors for 0 to n-1, but slot counter gets n activations
   if (i<SIGNAL_PULSES_IN_FRAME) {
      // open detector windows
      // gate SPD A
      qkdX::SpdMsg* gateSPDA = new qkdX::SpdMsg("TAGateSPDA", qkdX::token::SpdGateActive);
      gateSPDA->setDuration(spdGateDuration);
      sendDelayed(gateSPDA, offset + bobPathDiffDelay, "electricalGateA");
      // gate SPD D
      qkdX::SpdMsg* gateSPDD = new qkdX::SpdMsg("TAGateSPDD", qkdX::token::SpdGateActive);
      gateSPDD->setDuration(spdGateDuration);
      sendDelayed(gateSPDD, offset + bobPathDiffDelay, "electricalGateD");
      // gate SPD H
      qkdX::SpdMsg* gateSPDH = new qkdX::SpdMsg("TAGateSPDH", qkdX::token::SpdGateActive);
      gateSPDH->setDuration(spdGateDuration);
      sendDelayed(gateSPDH, offset, "electricalGateH");
      // gate SPD V
      qkdX::SpdMsg* gateSPDV = new qkdX::SpdMsg("TAGateSPDV", qkdX::token::SpdGateActive);
      gateSPDV->setDuration(spdGateDuration);
      sendDelayed(gateSPDV, offset, "electricalGateV");
   }
   // increment slot counter
   cMessage* incSlotCounterMsg = new cMessage("incSlotCounter", TA_INC_SLOT_COUNTER);
   scheduleAt(simTime() + offset, incSlotCounterMsg);
   // increment offset
   const double signalToSignalDelay = par("signalToSignalDelay");
   offset += (signalToSignalDelay + setupDelay + polarizationDelay);
}
```

Figure 26. Alice's Quantum Module Timing Parameters in Bob's Timing Analyzer Code.

Experimentation

Two experiments were planned and executed through the course of the research study. In each case, the experiment was shaped by operational parameters extracted from the research and systems reviewed in Chapters II and IV. The author designed and submitted the first experiment, known as the Beam Splitter Experiment, as a project for a course taken at the AFIT. Refer to Appendix A. Beam Splitter Experiment (BSE) for details. The second experiment, known as the Decoy State Enabled QKD Experiment, verified the performance of the decoy state enabled QKD model and attempted to validate it by comparing it to an operational system. The majority of the PNS Attack Experiment results are captured in Chapter V. Results from Beam Splitter Experiment that were not captured previously are discussed in the next section. Likewise, supplement results from the Decoy State Enabled QKD Experiment follow that discussion.

Supplement Beam Splitter Experiment Results and Findings

The process and results of the initial beam splitter experiment solidified understanding of the decoy state protocol for the research team, performed verification of the subsystem, validation of the notional architecture, and taught the author how to efficiently conduct larger experiments than had been accomplished using the notional architecture. The results related to V&V were captured in the modeling decoy state protocol journal article in Chapter IV. The results related to understanding the decoy state protocol are examined next and are followed by the discussion of efficiently conducting large experiments.

Understanding the Decoy State Protocol

The Beam Splitter Experiment was based on the research team's idea to use model Eve as a beam splitter. This translated into to adjusting the length of the fiber optic channel between Alice and Bob to proportionally model the loss induced by the subject beam splitter. This experiment employed the use of the notional architecture with minor modifications. The experiment process was partially automated through the use of Perl and shell scripting and distributed among computing resources available at AFIT.

The results of this experiment are partially reported in Chapter IV, in particular in the sections IV and V which discuss verification, validation and decoy state protocol considerations. In particular, the results of this experiment illustrate that a beam splitter cannot be used to accurately model a PNS attack. By employing a beam splitter, the photon number dependent yields of the signal and decoy states are, as expected, uniformly reduced between a "clean" system with no Eve and a system in which

Eve is represented as a beam splitter. Figure 27 and Figure 28 illustrate this result. In both figures, the estimated signal and decoy yields are plotted as green and black circles to represent those values in the absence of Eve and red and blue when Eve was considered to be present. Though the scales of the vertical axes are different, one can observe that loss due to Eve is approximately 50% for both the 1- and 2-photon yields for both the signal and decoy states. In a PNS attack, the estimated 1-photon yield should be nearly zero. In terms of the experiment, the results are inconclusive with respect to the null hypothesis because we cannot accurate model the PNS attack using a beam splitter.

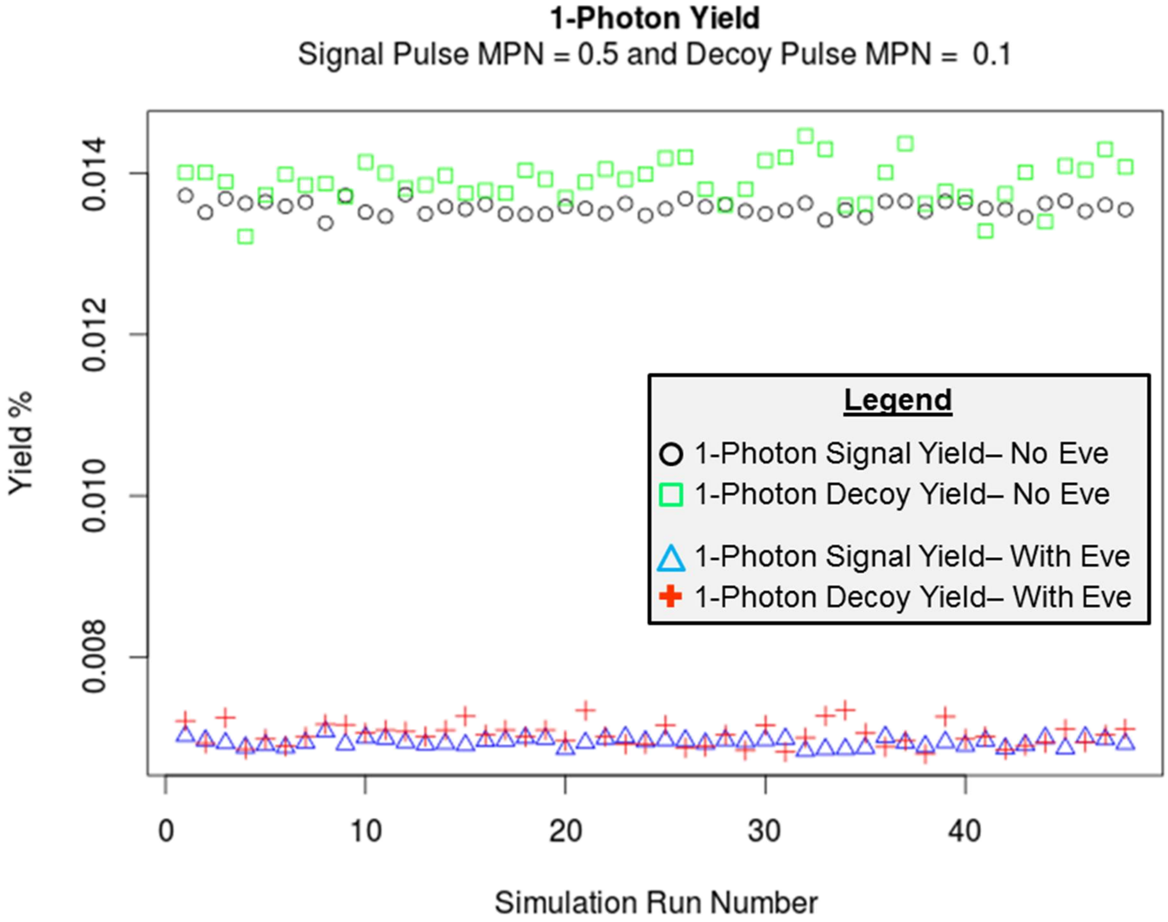


Figure 27. 1-Photon Signal and Decoy Yield With and Without Eve.

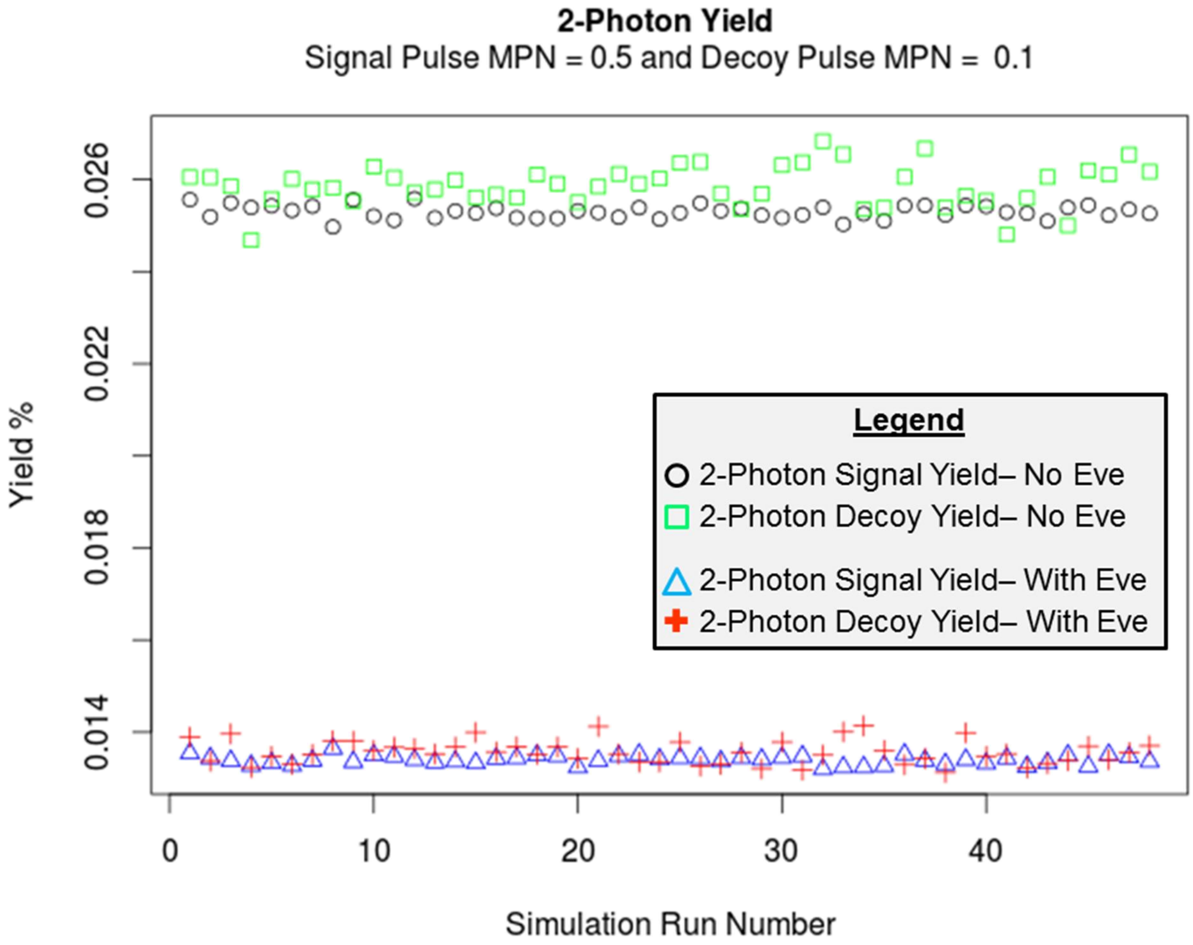


Figure 28. 2-Photon Yield and Decoy Yield With and Without Eve.

Additionally, we learned what it means to practically measure, or rather estimate, the photon number dependent yields and how a system can use this information. In a practical system which does not employ photon-number resolving detectors, the yield cannot be measured directly; the yield can only be estimated from the gain measurement. Equation Photon number dependent signal yield (12) captures this result and is recreated below for reference. As illustrated, yield is calculated based on the dark count rate, Y$_0$, the mean photon number, μ, gain, Q_μ, and the photon number of interest, n.

$$Y_n \cong Y_0 + 1 - \left(1 - \left[\frac{-\ln\left|1 + Y_0 - Q_\mu\right|}{\mu}\right]\right)^n. \qquad \text{Photon number dependent signal yield (12)}$$

Furthermore, the researcher recognized in this experiment that randomness in the model, i.e. noise in a practical system, will prevent the decoy state security condition from being true without adding some margin of error, i.e. tolerances. Such tolerances can be obtained from calibration of a known “clean” system in which Eve is not present. This notion is consistent with Lo *et al.’s* assertion that if Alice and Bob know their channel properties well, it strengthens their ability to use the decoy state protocol to detect the PNS attack [28]. This awareness urged our implementation to include and implement tolerances.

Executing Larger Experiments

Prior to the beam splitter experiment, the research team’s simulation runs generally consisted of running one simulation on a multiple processor/multi-core machine at a time. Single simulations could take hours or the better part of a day. Afterward, results were manually organized and derived from the standard output command line window or the graphical environment.

In contrast to this approach, the beam splitter experiment was planned to execute 96 treatments with a minimum of 30 trials to obtain statistically significant results. If one estimates that each run would take 4 hours, this experiment would take 1.3 years to complete if run sequentially on a single computer system. Essentially, the problem consisted of two parts: 1) untimely serial processing and 2) inadequate data recording and collection for this type of experimentation. The researcher was driven to solve this problem for himself and the team.

Initially, researcher discovered that engaging the multiple processors and cores could be accomplished by running multiple simulations in different command windows. However, data collection was still a problem. Once the experimental design defined what data was desired, the model was modified to output this information in a method that was more conducive to parallel simulation. Once this was accomplished, control scripts were created to:

1) Read the simulation configuration variables from a text file,

2) Modify the simulation source code,
3) Compile the code,
4) Adjust the random number seed between runs,
5) Execute the simulation, and
6) Organize the data.

With this process in place, execution was expanded to three high power workstations and 31 servers which could process 464 simulations simultaneously thereby reducing the time required to complete the experiment to just over 24 hours.

Further investigation exploited the inherent simulation capabilities of OMNeT++ and eliminated the need for the control scripts. However, this required the conversion of more than a dozen hard-coded simulation parameters into OMNeT++ NED file parameters. At completion of this research project, 77 QKD-specific simulation control variables are managed via a single configuration file.

Supplemental Decoy State Enabled QKD Experiment Results

The Decoy State Enabled QKD Experiment objectives included verification of the decoy state protocol's implementation and validation of the enhanced architecture. The results related to V&V were captured in the application of MBSE to QKD system article in Chapter V. Additional results not covered in this article, trial run findings, baseline signal gain, establishment of n-photon dependent yield thresholds, and experimental yield estimates.

Trial Run Findings

Two findings were observed during the trial runs of this experiment. The first finding caused a change to the model which involved the calculation of gain. The second finding enabled larger simulation runs to be executed.

Gain Definition

During analysis of experimental trial data, the researcher discovered that definition of gain varied in the surveyed research. As noted in Chapter II, initial decoy state research defined gain in terms of sifted bits, while later efforts defined it in terms of total detections because the sifting was extracted to a separate protocol efficiency variable "q". In the initial implementation of the decoy state protocol, we calculated gain based on the number of sifted bits. However, most researcher groups now define gain in terms of total detections. In terms of our model, the gain equation is:

$$Q_\mu = \frac{Number\ of\ signal\ detections\ at\ Bob}{Number\ of\ signal\ pulses\ sent\ by\ Alice} \qquad \text{Signal gain (practical measurement)} \qquad (20)$$

Thus, we had to make a small change to the software. This finding was documented in the *Detect Eavesdropper* use case and clarified in Chapter II as a result.

Simulation Configuration Finding

The second finding occurred when the research noticed unexpectedly high numbers of vacuum state detections at Bob during some longer simulation runs. The numbers measured exceeded the probability of the dark count detections configured by the simulation which suggested a problem. Upon running an even longer simulation, the simulation crashed with a segmentation fault which suggests a problem related to the program's memory space. At this point, the researcher realized the maximum number of frames, which controls the memory size of a series of arrays set at compile time, had been grossly exceeded. By adjusting this value and recompiling the simulation, the problem was solved.

Through some trial and error, the author found that maximum number of frames affects, i.e., slows, simulation performance when set above 80,000. Optimal performance can be obtained by selecting a value approximately 20% larger than an initial simulation run under current configuration conditions. Thus a trial run prior to any large set of experimental trials is highly recommended.

Signal Gain

After running the simulation 200 times, the mean value of the signal gain was 7.63E-3 and illustrated as a box plot in Figure 6. This value is on the same order of magnitude as the signal gain reported by Chen *et al.* [94]. This result established a means of comparison between the model and the previously reported results.

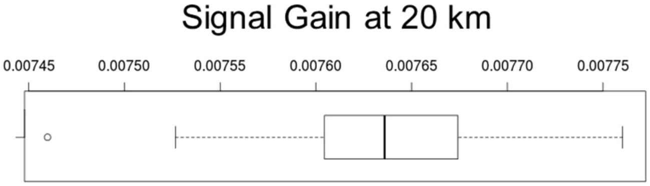


Figure 29. Distribution of Measured Signal Gain After 200 Trials.

Tolerances

In addition to the main objective of the experiment, the baseline, i.e. trials that did not include the PNS attack, established the normal operating thresholds, i.e., tolerances, for differences between the signal and decoy photon number dependent yields. These tolerances are defined by:

$$\left|Yield_n^{Signal} - Yield_n^{Decoy}\right| = Tolerance_n$$

Photon number dependent yield tolerance (21)

Individual tolerances were established for 1-, 2-, 3-, 4-, and 5- photon signal and decoy yield differences and are shown in Table 4.

Table 4. Experimentally Obtained Tolerances

n-Photon Signal and Decoy Difference Tolerance	**Value**
$Tolerance_1$	0.00211
$Tolerance_2$	0.00418
$Tolerance_3$	0.00621
$Tolerance_4$	0.00819
$Tolerance_5$	0.01012

n-Photon Yield Estimates

Similar to the Beam Splitter Experiment, the n-photon yield estimates for the results of this experiment were plotted as shown in Figure 30 through Figure 34. In each figure, the green and black points illustrate the respective signal and decoy yields for the baseline configuration in which Eve was not present to perform the PNS attack. In contrast, the blue and red points indicate the signal and decoy yields during the PNS attack. There is a noticeable difference yields measured when Eve performed the PNS attack. This is true all of the n-photon yields estimated in this experiment and is consistent with the simulation's ability to detect the PNS attack on all occasions. In terms of the experiment, we fail to reject the null hypothesis based on these results.

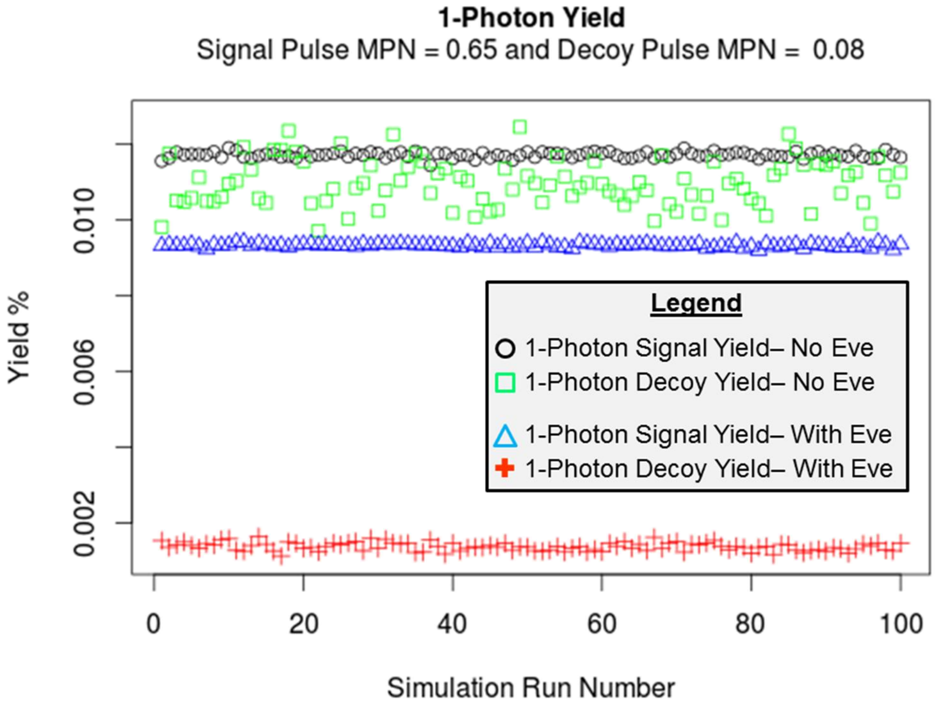


Figure 30. 1-Photon Signal and Decoy Yield With and Without Eve

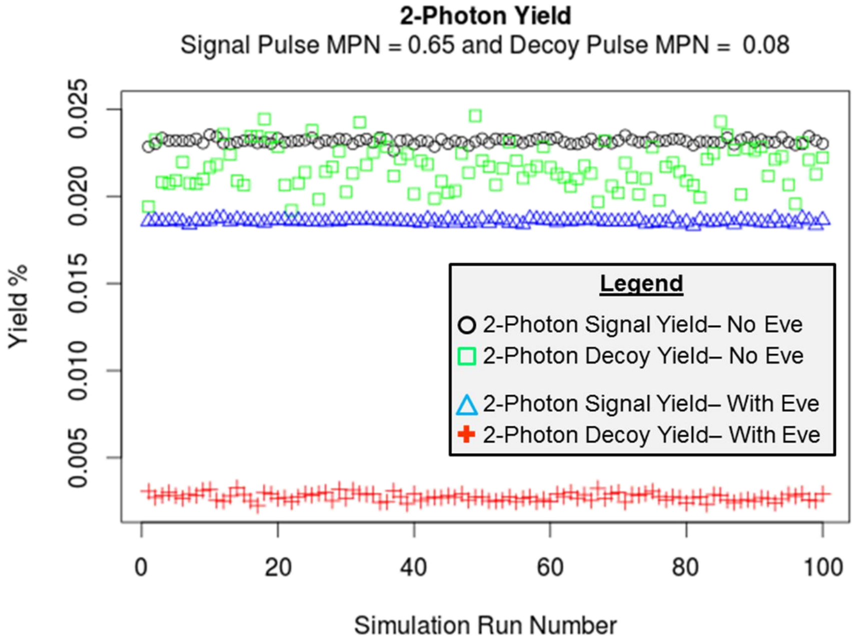


Figure 31. 2-Photon Signal and Decoy Yield With and Without Eve

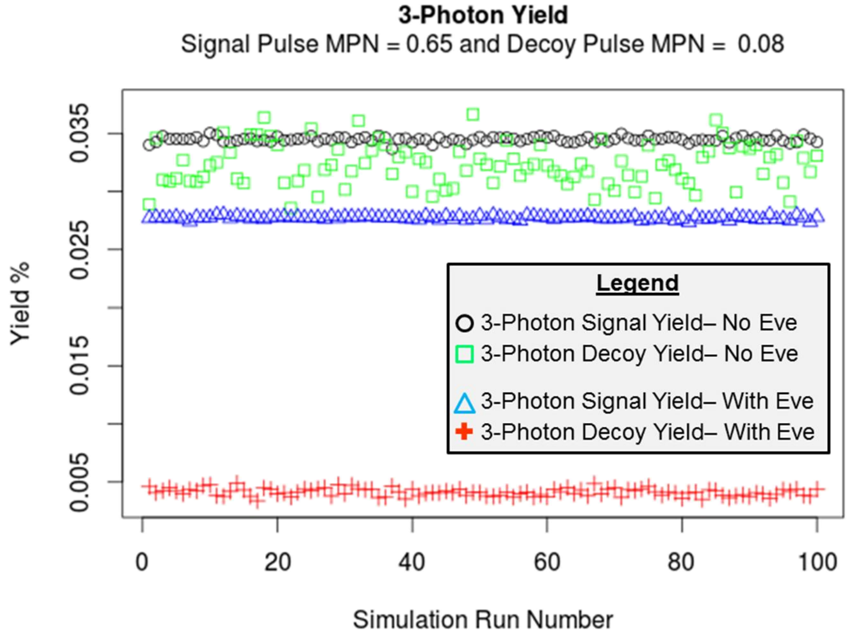


Figure 32. 3-Photon Signal and Decoy Yield With and Without Eve

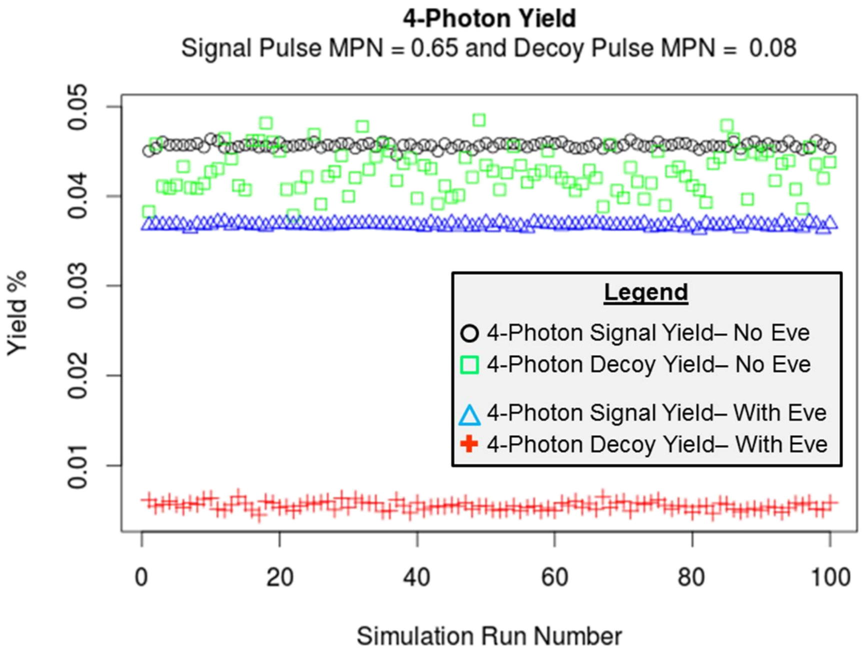


Figure 33. 4-Photon Signal and Decoy Yield With and Without Eve

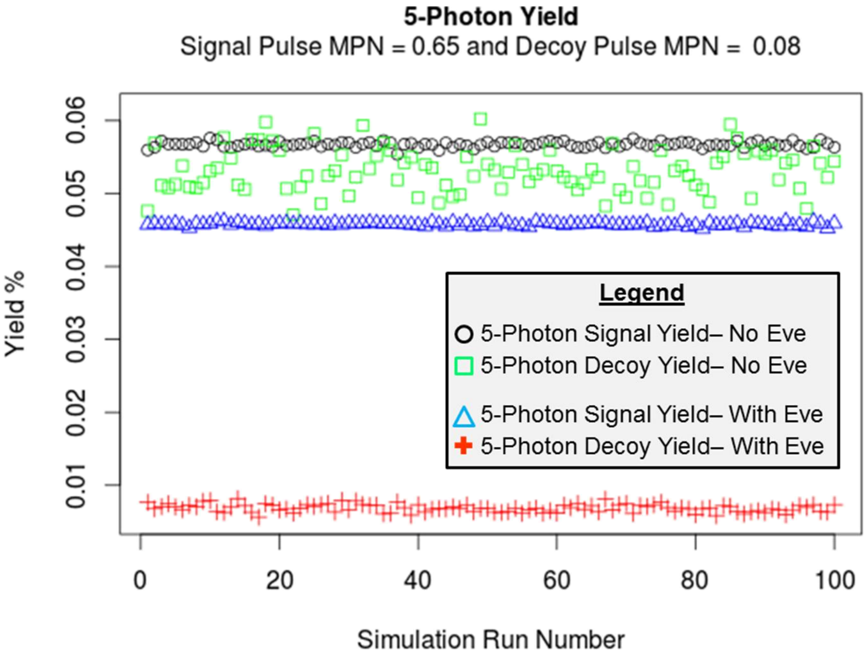


Figure 34. 5-Photon Signal and Decoy Yield With and Without Eve

Summary

This chapter discussed products and analyses that were not provided in the journal papers included as Chapters IV and V. The modeling development process was discussed as outlined in Chapter III as well as the design decision to reuse the existing notional architecture. Additionally, supplemental DoDAF OV-5b artifacts and use cases were presented. Next, two implementation findings, related to the decoy state protocol design and timing synchronization, were described. Lastly unpublished results from the Beam Splitter and Decoy State Enabled QKD Experiments were disclosed. Ultimately, this chapter explained how the research methodology was executed and closed the gaps on research results not yet published.

VII. Conclusion

Relevance of the Current Investigation

The overall goal of research was to develop an improved understanding of the operation and performance of the QKD decoy state protocol and provide a working model. The primary objective of this research was the design, implementation, verification, validation, and performance analysis of a decoy state enabled QKD system model using Systems Engineering (SE) Processes, Methods, and Tools (PMTs). QKD is an emerging cryptographic technology designed to provide unconditionally secure key distribution in applications where high levels of secrecy are required such as those in banking, government, and military environments. However, inherent complexities and limitations in practical implementations have prevented widespread adoption. Since the Systems Engineering methodology was intended to be applied to complex system solutions, it was selected to guide the research in this study.

This chapter summarizes the answers to all research questions presented in Chapter I. Next, the lessons learned while conducting the research are discussed. Finally, future research is proposed.

Answers to Research Questions

This study was guided by three primary research questions which evolved and expanded as the investigation progressed. Each question will be restated with its answer summarized in the following sections.

RQ1. What is Model-Based Systems Engineering (MBSE) and why is it important?

Model-Based Systems Engineering is an emerging methodology which integrates the use of models and modeling principles into various processes encompassed by the field of SE [50]. The SE methodology is a requirements-driven approach to developing and supporting multi-disciplinary systems from initial concept to disposal. Among the benefits of Systems

Engineering are reduced risk, lowered total life cycle cost, improved quality, and increased productivity [101, 33, 54, 25, 36].

RQ1.1 What MBSE Processes, Methods, and Tools (PMTs) are useful to a modeler in defining, implementing, verifying and validating a QKD system model?

A combination of the Vee and incremental development models found in the Banks *et al.*, Law and Kelton, and GUIDEx simulations steps provides a set of processes and methods for modelers to identify, define, decomposition, verify and validate system requirements and behaviors [25, 41, 33, 12, 16, 42, 7]. These processes and methods can be supported by tools such as DoDAF viewpoints, fully dressed use cases and diagrams, and spreadsheets which can be created and maintained using UMLet, EA, and programs found in the suite of desktop tools provided by Microsoft or Open Office [27, 12, 25, 67]. Furthermore, OMNeT++ and the Eclipse IDE provide a highly capable network communication simulation framework in which a QKD simulation was built.

RQ2. What is Quantum Key Distribution (QKD)?

Quantum Key Distribution (QKD) is the most mature application of the quantum information field and is touted as a revolutionary technology which offers the capability for two authenticated parties, Alice and Bob, to generate a symmetric secret key which can be used in conjunction with a symmetric encryption algorithm to enable secure communication. The nature of quantum mechanics enables the unique ability of a QKD system to detect the presence of an eavesdropper attempting to subvert the secure distribution of key material. The origins of QKD technology can be traced back to Stephen Wiesner, who first developed the idea of fraud-proof quantum money [75].

RQ2.1 What is the BB84 protocol?

Charles Bennett and Gilles Brassard operationalized this concept when they proposed the first QKD protocol (i.e., BB84) where single photons, representing quantum bits (qubits), are used to securely generate shared cryptographic key [2]. BB84 is a polarization- or phase-based prepare and measure QKD protocol defined by of eight phases of interactions between Alice and Bob as illustrated by Figure 12. First, Alice and Bob must authenticate their identities to each other. Transactional authentication may also be employed upon completion of the protocol for additional security [18]. Next, they perform the quantum exchange until Bob detects enough qubits to begin processing a secret key. The third phase, sifting, occurs when Alice and Bob compare their measurement bases to enable Bob to correctly interpret qubit values. The fourth phase is complete after Alice and Bob estimate the number of errors in this sifted key. In the next phase, they correct the errors in the sifted key. In the sixth phase, Alice and Bob estimate entropy, i.e. the amount of information leaked during the previous phases. Phase seven is known as privacy amplification and attempts to mitigate the losses due to the leaked information. The last phase occurs when Alice and Bob confirm they have matching keys by comparing a hash that each calculates [76].

RQ2.2 What is a Photon Number Splitting (PNS) attack?

As practical on-demand single photon sources are not available, most prepare and measure implementations attenuate a classical laser pulse down from millions of photons to a Mean Photon Number (MPN) on the order of 0.1. These sub-quantum pulses are often referred to as "weak coherent pulses" and are represented probabilistically by the Poisson distribution using MPN as the input parameter. The selection of the MPN is influenced by the losses in the optical channel and assumptions regarding the eavesdropper's technology [26] .

A low MPN results in poor qubit throughput and thus low key generation rates. However, a low MPN is often desirable to reduce the likelihood of multi-photon pulses, each of which exposes information about the distributed cryptographic key to an eavesdropper. This multi-photon security vulnerability has been subject of a number of attacks against the quantum channel. More specifically, the Photon Number Splitting (PNS) attack was conceived to gain maximum information from this vulnerability without detection. The theoretical PNS attack suggests that an all-powerful eavesdropper (Eve) can develop the same secret key without making her presence known to the authorized participants. She does so by blocking all single photons and stealing photons from the multi-photon pulses as they are in transit from the sender (Alice) to the receiver (Bob) [79, 19, 20, 21]. The decoy state protocol was introduced to mitigate the ability Eve to hide her presence during this attack [24].

RQ2.3 What is the Decoy State protocol?

The decoy state protocol represents a significant advancement in QKD technology, as the protocol is relatively cheap and easy to implement against the PNS vulnerability and can be configured to increase throughput on the quantum channel. The protocol was introduced in 2003 [24] and quickly improved in an initial series of works [28, 29, 82] and later [6, 83, 84, 85, 86, 87] by a group of researchers (supplemental works of note include [88, 89, 90, 91, 92]). In this three state configuration, the signal state, μ, facilitates higher key distribution rates and greater operational distances through an increased MPN, while the decoy state, ν, is used to increase the likelihood of detecting an attack using differential statistical analysis. Utilizing different MPNs for the signal and decoy states, allows the QKD system to detect photon number specific interference using the decoy state protocol security condition (described in the next section). The vacuum state is used to determine the detector error rate at Bob, where erroneous detections, known as "dark counts" are measured [93, 14].

Decoy state enabled QKD systems utilize the conventional signal state plus dedicated security states (decoy and vacuum). The signal states are used to transmit the qubits used for generating the shared secret key. The decoy states are used to increase the likelihood of detecting photon number specific interference, i.e., the PNS attack, on the quantum channel through statistical differentiation with the signal state. The vacuum states are used to determine the detector dark count rate when no photons are present. Alice randomly selects the type of state according to pre-determined occurrence percentages [14, 94].

RQ3. How effective is the Decoy State Protocol at detecting the Photon Number Splitting (PNS) attack?

The Decoy State Protocol was found to be effective in detecting a PNS attack in which Alice and Bob were able to characterize their system performance in the absence of a PNS attack. Of the 100+ simulations performed, using operational parameters derived from Chen *et al.'s Field Test of a Practical Secure Communication Network with Decoy-State Quantum Cryptography*, 100% successfully detected the PNS attack. However, a generalized inference to other Decoy State Enable QKD systems cannot be made as the Chen *et al.* configuration was neither a randomized experiment nor comprehensively representative of the population of QKD systems.

RQ3.1 What measurements are necessary to detect an eavesdropper performing a PNS attack?

The decoy state security condition can be evaluated using photon number dependent yields or error rates [28, 14]. The method involving yield was explored in this study, due to the low levels of system noise, and identified in Equation Decoy State Security Condition for Yields (1). The experiment conducted during this research explored the practical application of the condition and a derivation of Equation Decoy State Security Condition for Yields (1) which included a tolerance as defined by Equation Photon number dependent yield tolerance (21).

While error rates can be measured more directly, yields cannot be directly measured from current QKD systems. Yield estimates can be obtained from the MPN, the number of photons, the signal and decoy state gain measurements and the system dark count rate as shown by Equation Photon number dependent signal yield (12).

RQ3.2 What behaviors should be captured in a decoy state enabled QKD system model?

The "To-Be" DoDAF OV-5a and use cases identify the activities and behaviors which need to be captured in the decoy state protocol. These behaviors include the selection and announcement of a pulse state type, estimation of photon number dependent yields or QBER, and a comparison of the yields or QBER.

RQ3.3 How flexible is the qkdX framework for building and studying fit-for-purpose models?

Flexibility is somewhat difficult to quantify, however the qkdX framework was very capable of being adapted to support the decoy state enabled QKD system model. Yet this adaptation required a good deal of work by much of the research team and required the team members to have knowledge of both QKD system and software engineering.

The selection of OMNeT++ to support this framework strongly facilitated the ability of this project to produce a more configurable system model than in the past. During this project, the OMNeT++ simulation enabled nearly all of the configuration variables to be defined at run time through a single configuration file. This characteristic is more conducive to providing a flexible framework than one which requires a user to re-configure and re-compile the simulation for each change.

Some key characteristics of the system still remain cumbersome to configure even though improvements have been made. For example, timing synchronization and the MPN output at Alice, while improved, still require much attention from the user to change. This is an area which will be proposed for future research.

Overall, it is the author's assessment that this project has increased the flexibility of the qkdX framework and subsequent models by providing an opportunity to develop a simulation that answers a specific question, i.e. is fit-for-purpose.

RQ3.4 What changes need to be made to the existing qkdX simulation framework for studying a decoy state enabled QKD system model?

The OV-5a and use cases capture the behaviors which needed to be implemented to model the decoy state protocol. These changes were implemented in Alice and Bob's Processor, Blackboard, and messages used during Exchange Quantum Pulses/Bit, Generate Sifted Key, Estimate QBER, Reconcile Errors activities. The list of files requiring modification to implement the decoy state enabled BB84 protocol is as follows:

Table 5. qkdX Framework Changes for Decoy State Protocol

Area	**Files**
Alice	AliceBlackboard.cc AliceBlackboard.h AliceBlackboard.ned BasicAliceProcessor.cc BasicAliceProcessor.h BasicAliceProcessor.ned
Bob	BobBlackboard.cc BobBlackboard.h BobBlackboard.ned BasicBobProcessor.cc BasicBobProcessor.h BasicBobProcessor.ned
Messages	DetectionsABMsg_m.cc DetectionsABMsg_m.h DetectionsABMsg.msg ErrorBlocksPkt_m.cc ErrorBlocksPkt_m.h ErrorBlocksPkt.msg ErrorEstimationsMsg_m.cc ErrorEstimationsMsg_m.h ErrorEstimationsMsg.msg ErrorResultsMsg_m.cc ErrorResultsMsg_m.h ErrorResultsMsg.msg
Other	Blackboard.cc Blackboard.h Blackboard.ned

Additionally, changes which remove most of the hardcoded configuration parameters is also highly desirable. Comparisons of both Table 9 and Table 16 and both Table 10 and Table 17 highlight precisely where many of these changes occurred.

To implement the PNS attack, all of the optical component devices and modules in qkdX had to be modified to handle Fock state pulses to support this capability. This modification was designed and implemented primarily by another member of the research team, but the author did provide assistance during the late stages of implementation.

This concludes the summary and review of the research questions. A discussion of lessons learned follows.

Lessons Learned

This section will communicate the lessons learned by the researcher which he believes were relevant, but did not directly answer a research question, or relate to a research goal or objective. The most significant lesson learned involved problem solving. The author was able to apply it more than once.

Problem Solving

During analysis of the timing synchronization between Alice and Bob, the researcher learned the importance of clearly identifying and defining a problem before attempting to craft a solution. In this instance, he attempted to implement a change to the system that would make Alice and Bob less dependent on knowledge of each other without fully understanding the nature of the dependencies or clearly identifying the objective. The implementation attempt cost nearly two weeks of effort which did not produce successful results.

After a break, he reengaged the issue by starting with a more clear definition of the problem. Next he methodically collected data until he was certain that problem was well understood. Afterwards he developed a conceptual solution and measured timing data at points in the system to confirm his hypothesis. Once his hypothesis was confirmed he systematically made changes and performed additional tests to confirm the results. This systematic approach took him approximately two days and resulted in successful decoupling of timing knowledge between Alice and Bob.

While the latter approach might seem obvious to the reader as it is simply application of the scientific method. In his defense, the author warns that it is very easy to become lost in the details and over-confident that a particular solution will work before clearly understanding what problem needs to be solved. He also learned that taking a step back from the problem and its details was beneficial. It enabled him to assess the issue with a fresh set of eyes and to re-evaluate his assumptions.

This approach was applied again later when another researcher was struggling with implementing the PNS attack. The author was able to assist by remaining focused on systematically collecting data, testing hypotheses and when all else failed, he encouraged re-evaluating assumptions. His approach proved successful and again the problem was solved fairly rapidly.

Conclusions

Overall, the author found using MBSE methods, processes, and tools were invaluable to the success of the research project. The benefits facilitated clarification and communication of this complex system. For example, creating “As-Is” and “To-Be” architectures during model development versions enabled identification of capability gaps and focus the work effort in these areas. He also found that functional and activity decomposition captured during analysis and design translated directly into behavior modeling and algorithm implementation. Additionally, the employment of use cases increased the team’s understanding of the decoy state protocol and helped identify dependencies between phases. Furthermore, use cases provided a means to capture design decisions to answer *why* a particular approach was chosen. These artifacts improved communication of design decisions compared the processes and methods used in our previous work.

Additionally, the requirements-driven approach provided traceability between early and late phases of the study which permitted incremental verification and shaped the experiment design to support both system verification and validation. Using MBSE tools to capture requirements, design decisions, and identify verification methods greatly streamlined this process.

While there were some significant benefits to using this approach, the researcher also encountered a few minor issues which had to be overcome. The first issue arose while decomposing the BB84 and decoy state protocols. As discussed, the OV-5a is an activity oriented model in which the activities are defined by verb phrases. This caused some concern amongst team members who were accustomed to referring to the activities as gerunds, e.g., *Sifting* (gerund) vs. *Generate Sifted Key* (verb phrase) and *Quantum Exchange* (gerund) vs. *Exchange Quantum Bit*. Additionally, we stumbled somewhat while defining subsystem actors due to the propensity of actors to be human. As noted, we settled on identifying the protocol as the actor. However, this affinity for actors to remain human is noted in the use cases where the *Alice Protocol* and *Bob Protocols* actor names are shortened to *Alice* and *Bob*. Furthermore, we discovered that balancing diagram complexity and clear communication of ideas was an art. For example, system level use case diagrams contain much information but are difficult to read. We found that breaking this diagram into smaller pieces which focused on a particular area was beneficial. Despite these minor issues, the benefits of the MBSE approach paid great dividends in our research.

This paper presents the first known application of MBSE methodology to define and analyze a decoy state enabled QKD system. In this paper, we examined the use of MBSE PMTs to perform requirements elicitation, activity decomposition, system design, model verification, and validation activities to shape our decoy state enabled QKD modeling and simulation study. This study outlines the progression of a QKD notional architecture into a fit-for-purpose decoy state enabled QKD simulation tool.

By employing MBSE PMTs and experimental design, the author captured and performed verification and validation on a system model requirements. Specifically, he was able to show the decoy state enabled QKD system model was able to match, within acceptable limits, an empirical signal gain value identified in the literature. The analysis was used to develop evidence of the effectiveness of the decoy state enabled BB84 protocol to detect a PNS attack and enables further parameter studies of decoy state enabled QKD systems.

The results of the research highlight the value of using MBSE products, tailored for specific purposes, to conduct the analysis of complex systems and to facilitate communication of system architecture details to stakeholders. Overall, MBSE was an appropriate methodology for managing the overall engineering approach for a complex QKD system involving team members from multiple disciplines.

Future Research

This study has uncovered a variety of options and paths which are ripe for future research. A few notable opportunities are discussed in the following paragraphs.

This thesis focused on the yield comparisons defined by the decoy state security condition. However, the decoy state security condition allows for PNS detection through the comparison of signal and decoy state error rates as well. Thus, adding this error rate comparison to the model would be an opportunity for future research.

Additionally, automatic calibration of the mean photon number and internal timing delays would be a reasonable behavior for practical systems. Additionally, this capability would provide for a more flexible QKD simulation architecture. If the architecture were configured to calibrate itself, it would enable users to focus on studying behaviors more pertinent to analyzing the security-performance trade space of QKD systems. As it exists, a change in the architecture of Alice will require the user to make manual adjustments to re-calibrate Alice's timing and output MPNs for signal and decoy states.

In the existing framework, pseudo-randomness is provided by various random number generators (RNGs) employed by multiple system components. However these RNGs all use the same random number stream. Unfortunately, this induces correlation between elements which use random numbers, that is, the use of an RNG by one component will necessarily affect the use by another component. As OMNeT++ offers the capability to use multiple RNG streams, this area should be explored further.

While assessing the simulation results, one might notice the lack of variation in, for example the 1-photon yield as shown in Figure 30. This characteristic might be highly desirable in a practical implementation; it is less interesting to researchers who are attempting to study certain variations. Thus, future research should explore the decoy state protocol and exercise the capability of the qkdX framework to provide randomness in laser pulses and in the quantum channel.

Future research includes accomplishing sensitivity analysis by manipulating control variables as well as adding variability to the decoy state enabled QKD system model to further define the security-performance trade space. As only one configuration of a decoy state enabled QKD system was investigated, it was impractical to apply additional statistical methods, such as linear regression, in order to predict a (more) optimal system design. Varying more QKD system model input parameters would enable the research team to apply this method to explore and analyze more of the security-performance trade space of these systems.

Another opportunity for future research and verification of sub-system resides with the Single Photon Detectors (SPDs) in Bob. During the implementation process, we tracked the photon number (for Fock state pulses) and MPN (for "classical" weak coherent pulses) up to the detectors in Bob, but we did not verify what happened to the pulses inside of the detectors. The process of handling detections at Bob is a critical step for QKD systems and this is an area which should require special attention. At the conclusion of this study, the verification of the SPD's modeled behavior is beginning. If not planned, this endeavor should include verification that photons are handled appropriately.

Furthermore, some use cases were identified during the system analysis which were not defined or defined with few details during this study. Examples include *Authenticate Partners*, *Estimate Entropy*, and *Perform Privacy Amplification*. These use cases should be (further) defined if these activities are to be studied in future research.

One final proposal for future study was an early secondary goal for this research project. The DoD High Performance Computing Modernization Program (HPCMP) has resources available at the Air Force Research Lab (AFRL) which are available for no cost to research users at AFIT. Exploring a large set of security-performance trade spaces could presumably be accomplished in relatively short order by employing these resources.

Appendix A. Beam Splitter Experiment (BSE)

Introduction

This section describes the methodology used to conduct an experiment involving a QKD simulation framework. The experiment supports research that seeks to characterize the effectiveness, in terms of performance and security, of a decoy state enabled QKD protocol to detect the presence of an eavesdropper. In this research, security will be measured in terms of the system's ability to detect the presence of an eavesdropper during the quantum exchange phase of the key distribution protocol.

Objective

The ability to detect an eavesdropper using the differences of signal and decoy state yield measurements is purpose of the decoy state protocol and the goal of this experiment. Thus, this experiment will measure and compare differences between the yield of the signal and decoy states. The differences between the signal state and decoy state yields will be used to assess the ability of the model to detect an eavesdropper [24, 28]. In a real system or stochastic simulation, the difference is unlikely to be exactly zero. Thus, a pre-defined tolerance for each photon number specific yield estimate is used to define an acceptable interval within which the difference is expected to fall.

The validity of the simulation results is an on-going area of research. Additionally, the verification of modeled components was also on-going at the time this experiment was conducted [15]. However, the results of this experiment will be compared to generalized expected results based on the work of previous research and educated analysis [28, 29, 14, 6, 83, 89, 90, 92, 102]. It is not the intended purpose of this experiment to validate either the efforts of previous research or conclusively prove the validity of the simulation and its associated modeled system representation.

The hypotheses of the experiment are as follows:

1. H_0: $\left|Yield_n^{Signal} - Yield_n^{Decoy}\right| \leq Tolerance$
2. H_1: $\left|Yield_n^{Signal} - Yield_n^{Decoy}\right| > Tolerance$

The null hypothesis is based on the condition established by Lo *et al.* [28] and Equation Photon number dependent yield tolerance (21). This hypothesis will be based on the 1-, 2-, 3-, 4-, 5-photon yield estimates for this experiment.

System Boundaries

Figure 35 (OV-1) depicts a context view of a communications encryption system implementing QKD. In the figure, the boundary of the QKD model is illustrated by the outlined box labeled *Decoy State Enabled QKD System*. Clearly, the QKD system does not include encryption mechanisms nor is it dependent on what form of communication message will be encrypted.

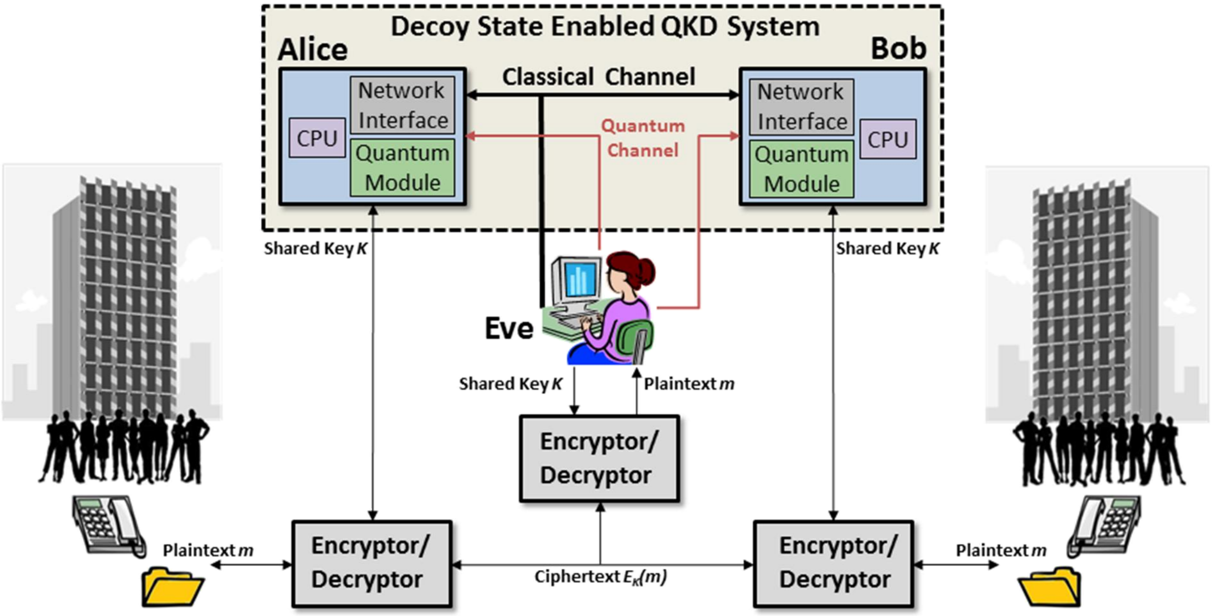


Figure 35. Operational Concept Diagram (OV-1).

Additionally, while the BB84 protocol consists of eight phases necessary for QKD, this experiment only needs to consider the results of the first five [15]. Particularly, the results from the quantum exchange and the error reconciliation phases are of most interest. From these phases many of the response variables will be recorded and measured.

There are two scenarios of interest that will be simulated. The first scenario consists of Alice and Bob connected by their classical and quantum channels with no other party on either channel. This scenario employs a polarization encoding scheme for qubits. In the second scenario, Eve is connected to both channels and passively sniffs 100% of the communications on the classic channel and actively intercepts a percentage of data on the quantum channel.

Experimental Limitations

The qkdX notional architecture does not contain the effect of an eavesdropper, Eve, executing a PNS attack. Thus, to inject and model Eve's actions on the quantum channel, we had to improvise. The research team theorized that generic Eve can be modeled using a beam splitter which splits the pulse energy into two directions. In other words, a beam splitter will reduce the amount of energy that reaches Bob's detectors. Another way to reduce the amount of energy reaching Bob is to increase the distance between Alice and Bob. Therefore, a simple attack from Eve can be modeled by increasing the distance between Alice and Bob. This abstraction is anticipated to reduce the decoy state method's ability to detect the presence of an eavesdropper. NOTE: The author may occasionally refer to the actual PNS attack as an "intelligent" attack to differentiate it from the simplified beam splitter-based, "non-intelligent" attack.

Additionally, the experiment was also somewhat limited by the amount of time required to execute simulations. With the detector efficiency set to 100%, at a range of 15km, and an MPN of 0.1, it took over 4 hours of real time for the simulation to process 50,000 detections. NOTE: the simulation time just passes six seconds. In contrast, a practical QKD system would need to transmit and process a number of pulses that is on the order of billions.

Assumptions

The following assumptions are made for this experiment:

1. General
    1.1. The laws of physics are not violated, i.e. the theory of quantum mechanics can be applied.
2. QKD

2.1. There is no active eavesdropping occurring on the classic channel.
  2.1.1. Eve does not perform active attacks on the classical channel.
  2.1.2. Eve is connected to the public channel and sniffs 100% of the packets transmitted without changing them.
2.2. Alice and Bob have been authenticated.
2.3. Alice and Bob are able to establish a baseline performance level of their quantum channel in which performance was not affected by Eve.
2.4. Any error in the pulse detected should be attributed to interference by Eve.
2.5. Eve's active interference is limited to the quantum channel.
2.6. The quantum channel is noiseless.
2.7. The quantum and classical channels are physically separated.
2.8. No attacks on the system are present against the QKD system other than Eve's active attack on the quantum channel and her passive listening on the public channel.
2.9. Reported dark count probabilities may exclude results which inadvertently caused Bob to measure a qubit during the gating period for a non-vacuum pulse type. Thus the actual value may be higher than the reported result but will be considered negligible due to the low chance of occurrence.
2.10. Reported after pulse probabilities may exclude results which inadvertently caused Bob to measure a qubit during the gating period for a non-vacuum pulse type. Thus the actual value may be higher than the reported result but will be considered negligible due to the low chance of occurrence.

3. Models
  3.1. The Poisson distribution is an accurate model of a practical photon number distribution.
    3.1.1. Photon numbers are independent.
  3.2. Detector efficiency is uniform and invariant.
  3.3. The delay between frames is invariant.
  3.4. The number of pulses per frame is invariant.
  3.5. Laser output is uniform and invariant.
  3.6. Attenuation in the quantum channel is uniform over any length of fiber.
  3.7. Signal to Signal delay is invariant.
  3.8. Temperature is uniform and invariant throughout the QKD system.
  3.9. The timing pulse rate is invariant.
  3.10. Polarization correction in Bob was 100% effective.
  3.11. Polarization drift did not occur in the quantum channel.
  3.12. The polarization reference frame was invariant.
  3.13. Optical components perform without degradation.
  3.14. Optical components are not damaged.

Response Variables

The response variables listed in Table 6 will be used to calculate the yield that was obtained during the quantum exchange phase of the BB84 protocol. The measurement method was excluded from the table as all response variables intended to be recorded and or output by the simulation application. A photon detection is counted when Bob's detectors are gated "open" and ready to detect a photon. A dark count is a spontaneous detection registered at Bob which did not result from a weak coherent pulse from Alice.

Table 6. Response Variables.

Response variable	Normal Operating Level	Measurement Precision	Relationship to Objective
# pulses sent	10e5 to 10e10	1 pulse	Contributes to signal & decoy yield
# signal states sent	% of total pulses sent	1 pulse	Contributes to signal yield
# decoy states sent	% of total pulses sent	1 pulse	Contributes to decoy yield
# vacuum states sent	% of total pulses sent	1 pulse	Contributes to dark count rate

Response variable	Normal Operating Level	Measurement Precision	Relationship to Objective
# signal states detected	% signals sent	1 pulse	Contributes to signal yield
# decoy states detected	% decoys sent	1 pulse	Contributes to decoy yield
# vacuum states detected	% vacuums sent	1 pulse	Contributes to dark count rate and yield
# sifted signals	50% of signal states detected	1 pulse	Contributes to signal yield
# sifted decoys	50% of decoy states detected	1 pulse	Contributes to decoy yield
# signals after error estimation	25% of sifted signals	1 pulse	Contributes to signal yield
# dark counts at Bob	(5 x 10^-6) of # gating periods at detector	1 pulse	Noise; contributes to yield

Control Variables

Table 7 contains the control variables or experimental factors that will be intentionally manipulated in order to produce differing output responses during the experiment. The variables are separated into additional categories, i.e., Percentages, Mean Photon Number, and Communication Range and Interference, for clarity. All control factors are used to account for both the effectiveness of the decoy state enabled QKD protocol as well as covariance effects. NOTE: the decoy and signal attenuation values refer to the electronically variable optical attenuator setting required to adjust the MPN at the output of Alice to the desired values.

In this table, the *Normal Operating Level* refers to the default value in the notional architecture which was designed to represent a generalized practical implementation. All control variables are numerical factors. The proposed settings for each variable were based on operational parameters to which the research team agreed.
Refer to the *Simulation Environment* section and Chapter V for additional discussion.

Table 7. Control Variables.

Variable/Factor	Normal Operating Level	Measurement Precision	Proposed Setting	Relationship to Response Variables (Predicted Effect)
Percentages				
% of decoy states	20	1	[30, 20, 10]	Affects magnitude of decoys detected and sent
% of signal states	70	1	[60, 70, 80]	Affects magnitude of signal detected and sent
Mean Photon Number				
Decoy Attenuation (dB)	17	0.1	[10.5, 13.5]	Controls Decoy MPN
Decoy State MPN	0.05	0.000 000 1	[0.1, 0.2]	Larger numbers increase magnitude of decoys detected
Signal Attenuation (dB)	14	0.1	[5.2, 6.6]	Controls Signal MPN
Signal State MPN	0.1	0.000 000 1	[0.5, 0.8]	Larger numbers increase magnitude of signals detected
Communication Range and Interference				
Distance (km)	10	0.001	[15, 50]	Larger distance decreases ratio of detections to pulses sent
Eve Aggressiveness	[0-100]	0.1	[0, 10, 25, 50]	Affects ability to detect Eve

Factors to hold constant are listed in Table 8 are a large subset of the total number of experimental factors that are held constant. These factors are somewhat independent of the particular simulation selected and have a non-negligible impact to a QKD system. Only the Error Correction Method and Protocol are categorical factors; the rest are numerical. NOTE: The error rate threshold is used by the protocol after the error rate is calculated. If the calculated error rate is greater than the specified threshold, the assumption is that an eavesdropper is interfering during quantum exchange and the protocol will terminate at this point.

Table 8. Factors to Hold Constant.

Variable/Factor	Desired Level or Allowable Range	Measurement Precision	Anticipated Effects on Responses or Objective
Dark count probability (%)	0.000 005	0.000 000 5	Larger probability will increase yield
Detector efficiency (%)	100	1	Larger percent will increase yield
Error correction method	decoyPerfect	N/A	Simplified error rates
Error rate threshold	0.25	0.005	Eavesdropper detection ability
Maximum # of frames	8,192	1	Actual simulation run time
Number of detections	50,000	1	Terminating condition for simulation
% of vacuum states	10	1	Contributes to dark count rate
Protocol	BB84	N/A	High security
Pulses per frame	1000	1	Affects simulation duration
Signal loss in fiber (dB/km)	0.2	0.05	Lower yield over distance

Nuisance Factors

Due to the fact that experiment will be conducted via computer simulation, the nuisance factors are negligible. That is, many of the nuisance factors are controlled by holding them constant or excluding them from the simulation.

Known/Suspected Interactions

The specified percentages and MPN for signal, decoy, and vacuum states are expected to affect the efficiency of the yield over greater distances and thus the likelihood of detecting an eavesdropper during the quantum exchange phase of the BB84 protocol.

Restrictions

Due to the nature of computer simulations, the experiment has few restrictions. For example, the simulated phenomena are programmed to obey the laws of physics and nature. Again by using computer simulation, it is relatively easy to adjust the control variables.

Two practical realities affect the results of the experiment: data and time. First, the author anticipates that any data required will be available, but possibly not in the most convenient format. Furthermore, he anticipates there will be a learning curve in how to evaluate the data the simulation produces. Second, the other large restriction will be proportional to the amount of time require to execute the test runs. As the control variables suggest, there are a large number of factors and levels that could be evaluated. Thus the experiment may take a lot of time to flesh out the relevant experimental regions from the irrelevant.

Design Preferences

The author anticipates that replication will be the most effective design principle to employ. Replication will enable the analyst to build confidence intervals and make comparisons between the observations. Blocking, in the sense of batch processing may be employed to conduct test runs. Randomization can be used to plan the order in which to adjust the control variables. Again, due to the nature of computer simulation, some principles of blocking and randomization are irrelevant.

Analysis & Presentation techniques

Analysis will involve the use of box plots to perform a preliminary visual examination of the output data. If the data appears to be within the range of expectations, confidence intervals on the difference of the estimated signal and decoy state yields will be calculated. These confidence intervals will be represented in a tabular format or graphically as necessary.

Coordination

Weekly teleconferences are normally conducted with the entire team and attended by the sponsor and SMEs. Additionally, coordination was required between the AFIT QKD team and AFIT's Center for Cyberspace Research in order to obtain additional computer hardware for simulation processing.

Necessity of Trial Runs

Trials runs are a necessity. The plan is as follows:

1. Conduct runs to verify the author's understanding of the configured system and BB84 protocol.
2. Conduct additional runs to verify that control variables are adjusted correctly by comparing the observations to expectations.
3. Develop control scripts to batch process the run.
4. Conduct additional trial runs as necessary.

Simulation Environment

The qkdX notional architecture will be configured and simulated on 31 Dell Servers and 2 HP Workstations. Most, i.e., 26 of the Dell servers have 12 cores each. The five remaining servers have 16 cores and the two workstations have 32 cores. Thus, with each core running a single simulation, 26*12 + 5*16 + 2*32 = 456 simulation runs can be executed simultaneously. Generally, each machine will run one treatment at a time with each core performing the simulation using a different random number seed.

Looking forward to the statistical analysis, it is statistically desirable to obtain the same number of data sets resulting from each treatment. Thus, each core in the 12-core machines will perform 4 replications and the 16-core machines will perform 3 replications. Therefore each machine will produce a total of 48 data sets per treatment.

Another noteworthy characteristic of the notional architecture's simulation environment is the locations of the configuration control variables and factors to hold constant. These variables are spread out across the framework in a design that is very well suited to object-oriented design but arguably less desirable for a flexible simulation environment. Table 9 outlines the locations of the control variables necessary to configure the notional architecture. Similarly, the file locations of the *Factors to Hold Constant* are identified in Table 10.

Table 9. Control Variable File Locations.

Simulation Variable	**qkdX File Location**
Percentages	
% of decoy states	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h
% of signal states	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h
Mean Photon Number	
Decoy Attenuation (dB)	qkdX/examples/qkdXDemo_v2014/src/alice/quantum_module/aliceQuantumModuleCtrl/models/BasicAliceQuantumModuleCtrl.cc
Decoy State MPN	qkdX/examples/qkdXDemo_v2014/src/alice/quantum_module/aliceQuantumModuleCtrl/models/BasicAliceQuantumModuleCtrl.cc
Signal Attenuation (dB)	qkdX/examples/qkdXDemo_v2014/src/alice/quantum_module/aliceQuantumModuleCtrl/models/BasicAliceQuantumModuleCtrl.cc

Signal State MPN	qkdX/examples/qkdXDemo_v2014/src/alice/quantum_module/aliceQuantumModuleCtrl/models /BasicAliceQuantumModuleCtrl.cc
Communication Range and Interference	
Distance (km)	qkdX/examples/qkdXDemo_v2014/simulations/Network_QKD.ned
Eve Aggressiveness	qkdX/examples/qkdXDemo_v2014/simulations/Network_QKD.ned

Table 10. File Locations for Factors to Hold Constant.

Variable/Factor	**qkdX Location**
Dark count probability (%)	qkdX/examples/qkdXDemo_v2014/src/bob/quantum_module/TimingAnalyzer.ned
Detector efficiency (%)	qkdX/examples/qkdXDemo_v2014/src/bob/quantum_module/TimingAnalyzer.ned
Error Correction Method	qkdX/examples/qkdXDemo_v2014/omnetpp.ini
Error rate threshold	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h
Maximum # of frames	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h
Number of detections	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h
% of vacuum states	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h
Protocol	qkdX/examples/qkdXDemo_v2014/omnetpp.ini
Pulses per frame	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h
Signal loss in fiber (dB/km)	qkdX/framework/qkdX/channels/FiberChannel.ned
Pulse duration	qkdX/framework/qkdX/models/pulse/PulseFactory.cc
SPD Gate Duration (ns)	qkdX/examples/qkdXDemo_v2014/src/bob/quantum_module/TimingAnalyzer.ned
Temperature (degrees C)	qkdX/framework/qkdX/channels/FiberChannel.ned
Timing pulse wavelength	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h
Weak coherent pulse wavelength	qkdX/examples/qkdXDemo_v2014/src/common/demo-config.h

The maximum number of frames will be set after trial runs of the experiment. This value affects the amount of memory consumed by the simulation for various internal variables. It has already been observed that increases in the order of magnitude for this variable translate to additional processing time. Once the trials runs have been completed, this variable will likely be set to a constant value and will probably not vary with treatments.

Simulation Control Variables

The mean photon number is achieved by attenuating a classical laser pulse down to the desired mean number of photons [15]. Hence both MPN and attenuation are listed, but only attenuation is directly set in the simulation.

As discussed in the limitations section, Eve's Aggressiveness is based upon the relationship between beam splitters and fiber channel length. Therefore, rather than indicating an actual control variable, Eve's Aggressiveness is converted to an additional length of fiber optic cable with a fixed attenuation of 0.2 dB per km. Using the optical power definition of dB, 10 * $\log_{10}$ (Power$_{out}$/Power$_{in}$), 3dB is equivalent to a 50% loss [103]. Thus to model Eve's aggressiveness at 50%, an additional 15 km of fiber is added to the quantum channel to simulate the interference due to Eve.

Treatments

Another way to view the control variables is shown in Table 11. Here the control variables are separated into levels and one can more easily see there are 4*2*2*2*3 = 96 combinations of interest in a practical full factorial design. The only configurations that have been eliminated were those that were nonsensical to perform, e.g. signal percent = 80, decoy percent = 30, vacuum percent = 10.

The Eve Aggressiveness level of 100/0 (passive/aggressive) defines a scenario in which Eve is not present. Additionally, this set of 24 treatments characterizes the baselines of response variables for comparison against response variables from other treatments where all other controls are held at the same level. For example, the treatment such that Eve Aggressiveness = 100/0, distance = 15km, MPN Signal = 0.5, MPN Decoy = 0.2, and Signal/Decoy/ Vacuum Percentages = 60/30/10 is a baseline for treatments where Eve Aggressiveness is 90/10, 75/25, and 50/50 and distance = 15km, MPN Signal = 0.5, MPN Decoy = 0.2, and Signal/Decoy/Vacuum Percentages = 60/30/10. Thus three comparisons can be made using this baseline. Figure 36 illustrates the relationships between the baselines and the other treatments.

Table 11. BSE Control Variable Summary.

Eve Aggressiveness	Distance	MPN Signal	MPN Decoy	Signal/Decoy/Vacuum Percentage
90/10	15km	0.5	0.1	60/30/10
75/25	50km	0.8	0.2	70/20/10
50/50				80/10/10
100/0				

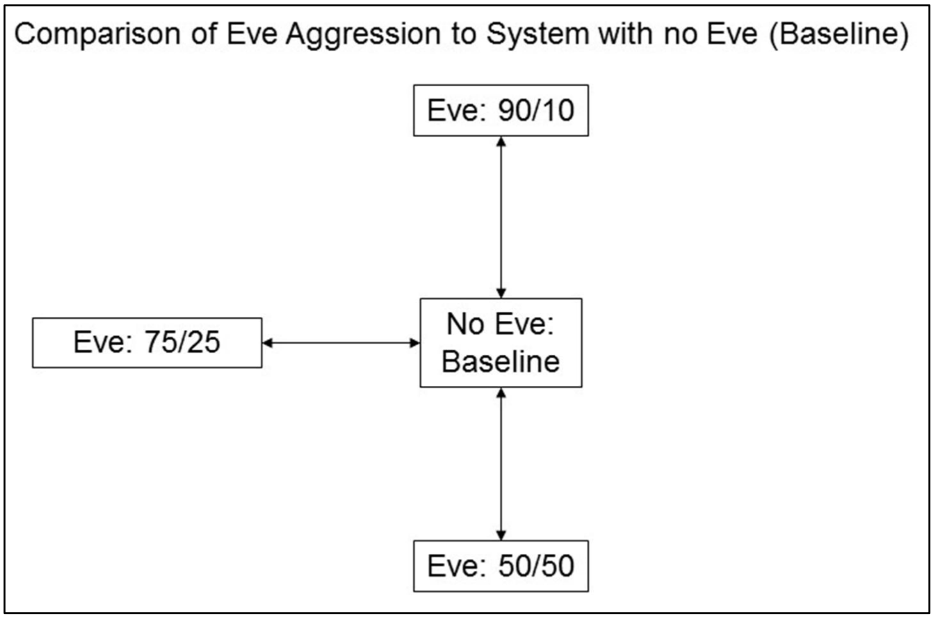

Comparison of Eve Aggression to System with no Eve (Baseline)
Eve: 90/10
Eve: 75/25
No Eve: Baseline
Eve: 50/50

Figure 36. BSE Baseline Comparisons.

The treatments are enumerated in Table 12. As discussed, the *Eve Aggressiveness* column identifies the percentage of the pulse energy lost across the quantum channel that is used to model Eve. Thus the *Effective Distance* reflects the original distance plus the additional length of fiber used to model Eve's aggressiveness.

Table 12. Test Matrix (List of Treatments).

Test #	Distance	Eve Aggress-iveness	Effective Distance	Signal State MPN	Decoy State MPN	Signal Atten-uation	Decoy Atten-uation	% Signal States	% Decoy States
1	15000	0	15000	0.5	0.1	6.6	13.5	60	30
2	15000	0	15000	0.5	0.2	6.6	10.5	60	30
3	15000	0	15000	0.8	0.1	5.2	13.5	60	30
4	15000	0	15000	0.8	0.2	5.2	10.5	60	30

Test #	Distance	Eve Aggress-iveness	Effective Distance	Signal State MPN	Decoy State MPN	Signal Atten-uation	Decoy Atten-uation	% Signal States	% Decoy States
5	15000	0	15000	0.5	0.1	6.6	13.5	70	20
6	15000	0	15000	0.5	0.2	6.6	10.5	70	20
7	15000	0	15000	0.8	0.1	5.2	13.5	70	20
8	15000	0	15000	0.8	0.2	5.2	10.5	70	20
9	15000	0	15000	0.5	0.1	6.6	13.5	80	10
10	15000	0	15000	0.5	0.2	6.6	10.5	80	10
11	15000	0	15000	0.8	0.1	5.2	13.5	80	10
12	15000	0	15000	0.8	0.2	5.2	10.5	80	10
13	50000	0	50000	0.5	0.1	6.6	13.5	60	30
14	50000	0	50000	0.5	0.2	6.6	10.5	60	30
15	50000	0	50000	0.8	0.1	5.2	13.5	60	30
16	50000	0	50000	0.8	0.2	5.2	10.5	60	30
17	50000	0	50000	0.5	0.1	6.6	13.5	70	20
18	50000	0	50000	0.5	0.2	6.6	10.5	70	20
19	50000	0	50000	0.8	0.1	5.2	13.5	70	20
20	50000	0	50000	0.8	0.2	5.2	10.5	70	20
21	50000	0	50000	0.5	0.1	6.6	13.5	80	10
22	50000	0	50000	0.5	0.2	6.6	10.5	80	10
23	50000	0	50000	0.8	0.1	5.2	13.5	80	10
24	50000	0	50000	0.8	0.2	5.2	10.5	80	10
25	15000	5050	30000	0.5	0.1	6.6	13.5	60	30
26	15000	5050	30000	0.5	0.2	6.6	10.5	60	30
27	15000	5050	30000	0.8	0.1	5.2	13.5	60	30
28	15000	5050	30000	0.8	0.2	5.2	10.5	60	30
29	15000	5050	30000	0.5	0.1	6.6	13.5	70	20
30	15000	5050	30000	0.5	0.2	6.6	10.5	70	20
31	15000	5050	30000	0.8	0.1	5.2	13.5	70	20
32	15000	5050	30000	0.8	0.2	5.2	10.5	70	20
33	15000	5050	30000	0.5	0.1	6.6	13.5	80	10
34	15000	5050	30000	0.5	0.2	6.6	10.5	80	10
35	15000	5050	30000	0.8	0.1	5.2	13.5	80	10
36	15000	5050	30000	0.8	0.2	5.2	10.5	80	10
37	50000	5050	65000	0.5	0.1	6.6	13.5	60	30
38	50000	5050	65000	0.5	0.2	6.6	10.5	60	30
39	50000	5050	65000	0.8	0.1	5.2	13.5	60	30
40	50000	5050	65000	0.8	0.2	5.2	10.5	60	30
41	50000	5050	65000	0.5	0.1	6.6	13.5	70	20
42	50000	5050	65000	0.5	0.2	6.6	10.5	70	20
43	50000	5050	65000	0.8	0.1	5.2	13.5	70	20

Test #	Distance	Eve Aggress-iveness	Effective Distance	Signal State MPN	Decoy State MPN	Signal Atten-uation	Decoy Atten-uation	% Signal States	% Decoy States
44	50000	5050	65000	0.8	0.2	5.2	10.5	70	20
45	50000	5050	65000	0.5	0.1	6.6	13.5	80	10
46	50000	5050	65000	0.5	0.2	6.6	10.5	80	10
47	50000	5050	65000	0.8	0.1	5.2	13.5	80	10
48	50000	5050	65000	0.8	0.2	5.2	10.5	80	10
49	15000	7525	21200	0.5	0.1	6.6	13.5	60	30
50	15000	7525	21200	0.5	0.2	6.6	10.5	60	30
51	15000	7525	21200	0.8	0.1	5.2	13.5	60	30
52	15000	7525	21200	0.8	0.2	5.2	10.5	60	30
53	15000	7525	21200	0.5	0.1	6.6	13.5	70	20
54	15000	7525	21200	0.5	0.2	6.6	10.5	70	20
55	15000	7525	21200	0.8	0.1	5.2	13.5	70	20
56	15000	7525	21200	0.8	0.2	5.2	10.5	70	20
57	15000	7525	21200	0.5	0.1	6.6	13.5	80	10
58	15000	7525	21200	0.5	0.2	6.6	10.5	80	10
59	15000	7525	21200	0.8	0.1	5.2	13.5	80	10
60	15000	7525	21200	0.8	0.2	5.2	10.5	80	10
61	50000	7525	56200	0.5	0.1	6.6	13.5	60	30
62	50000	7525	56200	0.5	0.2	6.6	10.5	60	30
63	50000	7525	56200	0.8	0.1	5.2	13.5	60	30
64	50000	7525	56200	0.8	0.2	5.2	10.5	60	30
65	50000	7525	56200	0.5	0.1	6.6	13.5	70	20
66	50000	7525	56200	0.5	0.2	6.6	10.5	70	20
67	50000	7525	56200	0.8	0.1	5.2	13.5	70	20
68	50000	7525	56200	0.8	0.2	5.2	10.5	70	20
69	50000	7525	56200	0.5	0.1	6.6	13.5	80	10
70	50000	7525	56200	0.5	0.2	6.6	10.5	80	10
71	50000	7525	56200	0.8	0.1	5.2	13.5	80	10
72	50000	7525	56200	0.8	0.2	5.2	10.5	80	10
73	15000	9010	17300	0.5	0.1	6.6	13.5	60	30
74	15000	9010	17300	0.5	0.2	6.6	10.5	60	30
75	15000	9010	17300	0.8	0.1	5.2	13.5	60	30
76	15000	9010	17300	0.8	0.2	5.2	10.5	60	30
77	15000	9010	17300	0.5	0.1	6.6	13.5	70	20
78	15000	9010	17300	0.5	0.2	6.6	10.5	70	20
79	15000	9010	17300	0.8	0.1	5.2	13.5	70	20
80	15000	9010	17300	0.8	0.2	5.2	10.5	70	20
81	15000	9010	17300	0.5	0.1	6.6	13.5	80	10
82	15000	9010	17300	0.5	0.2	6.6	10.5	80	10

Test #	Distance	Eve Aggress-iveness	Effective Distance	Signal State MPN	Decoy State MPN	Signal Atten-uation	Decoy Atten-uation	% Signal States	% Decoy States
83	15000	9010	17300	0.8	0.1	5.2	13.5	80	10
84	15000	9010	17300	0.8	0.2	5.2	10.5	80	10
85	50000	9010	52300	0.5	0.1	6.6	13.5	60	30
86	50000	9010	52300	0.5	0.2	6.6	10.5	60	30
87	50000	9010	52300	0.8	0.1	5.2	13.5	60	30
88	50000	9010	52300	0.8	0.2	5.2	10.5	60	30
89	50000	9010	52300	0.5	0.1	6.6	13.5	70	20
90	50000	9010	52300	0.5	0.2	6.6	10.5	70	20
91	50000	9010	52300	0.8	0.1	5.2	13.5	70	20
92	50000	9010	52300	0.8	0.2	5.2	10.5	70	20
93	50000	9010	52300	0.5	0.1	6.6	13.5	80	10
94	50000	9010	52300	0.5	0.2	6.6	10.5	80	10
95	50000	9010	52300	0.8	0.1	5.2	13.5	80	10
96	50000	9010	52300	0.8	0.2	5.2	10.5	80	10

Appendix B. Decoy State Enabled QKD Experiment

Introduction

This section describes the methodology used to conduct an experiment involving the qkdX simulation framework. The experiment supports research that seeks to characterize the effectiveness, in terms of performance and security, of a decoy state enabled QKD protocol to detect the presence of an eavesdropper. In this research, security will be measured in terms of the system's ability to detect the presence of an eavesdropper during the quantum exchange phase of the key distribution protocol.

Objective

The ability to detect an eavesdropper using the differences of signal and decoy state yield measurements is the purpose of the decoy state protocol and the goal of this experiment. Thus, this experiment will measure and compare differences between the yield of the signal and decoy states in the presence of a PNS attack. The differences between the signal state and decoy state yields will be used to assess the ability of the model to detect an eavesdropper [24, 28]. In a real system or stochastic simulation, the difference is unlikely to be exactly zero. Thus, a pre-defined tolerance for each photon number specific yield estimate is used to define an acceptable interval within which the difference is expected to fall.

The validity of the simulation results is an on-going area of research. Additionally, the verification of modeled components was also on-going at the time this experiment was conducted [15]. It is not the intended purpose of this experiment to validate either the efforts of previous research. However, the results of this experiment will be compared to generalized expected results based on the work of previous research and educated analysis. Furthermore, this experiment is intended to validate the simulation by comparing its results to those of other research [28, 29, 14, 6, 83, 89, 90, 92, 102].

The hypotheses of the experiment are as follows:

1. $H_0\colon \left|Yield_n^{Signal} - Yield_n^{Decoy}\right| \leq Tolerance$
2. $H_1\colon \left|Yield_n^{Signal} - Yield_n^{Decoy}\right| > Tolerance$

The null hypothesis is based on the condition established by Lo *et al.* [28] and Equation Photon number dependent yield tolerance (21). This hypothesis will be based on the 1-, 2-, 3-, 4-, 5-photon yield estimates for this experiment.

System Boundaries

Figure 37 (OV-1) depicts a context view of a communications encryption system implementing QKD. In the figure, the boundary of the QKD model is illustrated by the outlined box labeled *Decoy State Enabled QKD System*. Clearly, the QKD system does not include encryption mechanisms nor is it dependent on what form of communication message will be encrypted.

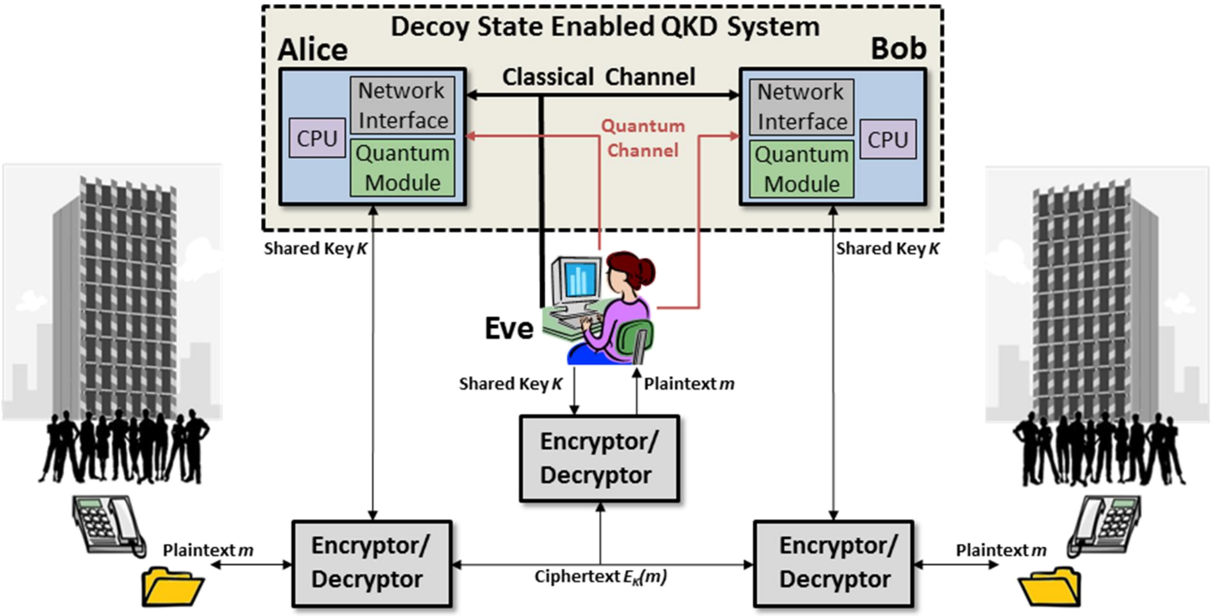


Figure 37. Operational Concept Diagram (OV-1).

Additionally, while the BB84 protocol consists of eight phases necessary for QKD, this experiment only needs to consider the results of the first five [15]. Particularly, the results from the quantum exchange and the error reconciliation phases are of most interest. From these phases many of the response variables will be recorded and measured.

There are two scenarios of interest that will be simulated. The first scenario consists of Alice and Bob connected by their classical and quantum channels with no other party on either channel. This scenario employs a polarization encoding scheme for qubits. In the second scenario, Eve is connected to both channels and passively sniffs 100% of the communications on the classic channel and actively intercepts a percentage of data on the quantum channel.

Experimental Limitations

This experiment will involve the research team's preliminary attempt to model a PNS attack. The employed PNS model will be limited to the behaviors the research team believes will affect measureable results by the decoy state protocol. Other PNS attack behaviors, which would be necessary for an eavesdropper to gain practical knowledge about the key exchange and for the eavesdropper to hide her presence, were excluded from the model.

Additionally, the experiment was also somewhat limited by the amount of time required to execute simulations. With the detector efficiency set to 100%, at a range of 15km, and an MPN of 0.1, it took over 4 hours of real time for the simulation to process 50,000 detections. NOTE: the simulation time just passes six seconds. In contrast, a practical QKD system would need to transmit and process a number of pulses that is on the order of billions.

Assumptions

The following assumptions are made for this experiment:

1. General
    1.1. The laws of physics are not violated, i.e. the theory of quantum mechanics can be applied.
2. QKD
    2.1. There is no active eavesdropping occurring on the classic channel.
        2.1.1. Eve does not perform active attacks on the classical channel.
        2.1.2. Eve is connected to the public channel and sniffs 100% of the packets transmitted without changing them.
    2.2. Alice and Bob have been authenticated.

2.3. Alice and Bob are able to establish a baseline performance level of their quantum channel in which performance was not affected by Eve.
2.4. Any error in the pulse detected should be attributed to interference by Eve.
2.5. Eve's active interference is limited to the quantum channel.
2.6. The quantum channel is noiseless.
2.7. The quantum and classical channels are physically separated.
2.8. No attacks on the system are present against the QKD system other than Eve's active attack on the quantum channel and her passive listening on the public channel.
2.9. Eve does not attempt to hide her presence on the quantum channel.
2.10. Reported dark count probabilities may exclude results which inadvertently caused Bob to measure a qubit during the gating period for a non-vacuum pulse type. Thus the actual value may be higher than the reported result but will be considered negligible due to the low chance of occurrence.
2.11. Reported after pulse probabilities may exclude results which inadvertently caused Bob to measure a qubit during the gating period for a non-vacuum pulse type. Thus the actual value may be higher than the reported result but will be considered negligible due to the low chance of occurrence.

3. Models
3.1. The Poisson distribution is an accurate model of a practical photon number distribution.
3.1.1. Photon numbers are independent.
3.2. Detector efficiency is uniform and invariant.
3.3. The delay between frames is invariant.
3.4. The number of pulses per frame is invariant.
3.5. Laser output is uniform and invariant.
3.6. Attenuation in the quantum channel is uniform over any length of fiber.
3.7. Signal to Signal delay is invariant.
3.8. Temperature is uniform and invariant throughout the QKD system.
3.9. The timing pulse rate is invariant.
3.10. Polarization correction in Bob was 100% effective.
3.11. Polarization drift did not occur in the quantum channel.
3.12. The polarization reference frame was invariant.
3.13. Optical components perform without degradation.
3.14. Optical components are not damaged.

Response Variables

The response variables listed in Table 13 will be used to calculate the yield that was obtained during the quantum exchange phase of the BB84 protocol. The measurement method was excluded from the table as all response variables intended to be recorded and or output by the simulation application.

Table 13. Response Variables.

Response variable	Normal Operating Level	Measurement Precision	Relationship to Objective
signalY1	$Y_0 + \left[1 - \left(1 - \eta_\mu\right)^1\right]$	0.000 1	1-Photon Signal Yield
signalY2	$Y_0 + \left[1 - \left(1 - \eta_\mu\right)^2\right]$	0.000 1	2-Photon Signal Yield
signalY3	$Y_0 + \left[1 - \left(1 - \eta_\mu\right)^3\right]$	0.000 1	3-Photon Signal Yield
signalY4	$Y_0 + \left[1 - \left(1 - \eta_\mu\right)^4\right]$	0.000 1	4-Photon Signal Yield
signalY5	$Y_0 + \left[1 - \left(1 - \eta_\mu\right)^5\right]$	0.000 1	5-Photon Signal Yield

Response variable	Normal Operating Level	Measurement Precision	Relationship to Objective
decoyY1	$Y_0 + [1 - (1 - \eta_\nu)^1]$	0.000 1	1-Photon Decoy Yield
decoyY2	$Y_0 + [1 - (1 - \eta_\nu)^2]$	0.000 1	2-Photon Decoy Yield
decoyY3	$Y_0 + [1 - (1 - \eta_\nu)^3]$	0.000 1	3-Photon Decoy Yield
decoyY4	$Y_0 + [1 - (1 - \eta_\nu)^4]$	0.000 1	4-Photon Decoy Yield
decoyY5	$Y_0 + [1 - (1 - \eta_\nu)^5]$	0.000 1	5-Photon Decoy Yield

Control Variables

Table 14 contains the control variables or experimental factors that will be intentionally manipulated in order to produce differing output responses during the experiment. The variables are separated into additional categories, i.e., Percentages, Mean Photon Number, and Communication Range and Interference, for clarity. All control factors are used to account for both the effectiveness of the decoy state enabled QKD protocol as well as covariance effects.

In this table, the *Normal Operating Level* refers the range of values reported in the survey of decoy state research reported in Chapters II and IV. Eve's Presence is a categorical factor whereas the others are numerical. With the exception of the *Eve's Presence* factor, the proposed settings of the control variables are coupled together.

The proposed settings for each variable were based on parameters from the Chen *et al.* study [94]. Refer to the *Simulation Environment* section and Chapter V for additional discussion.

Table 14. Control Variables.

Variable/Factor	Normal Operating Level	Configured Setting	Predicted Effect
Signal State MPN (photons/pulse)	[0.1-1+]	[0.65]	Larger numbers increase magnitude of signals detected
Decoy State MPN (photons/pulse)	[0.05-1+]	[0.08]	Larger numbers increase magnitude of decoys detected
% of signal states	[50-100]	[75]	Affects magnitude of signal detected and sent
% of decoy states	[0-40]	[12.5]	Affects magnitude of decoys detected and sent
% of vacuum states	[0-20]	[12.5]	Affects magnitude of vacuum signals detected & dark count rate
Distance (km)	[15-75]	[20]	Larger distance decreases ratio of detections to pulses sent
Quantum Channel Signal Loss	[Varies by distance]	[5.6]	Larger distance decreases ratio of detections to pulses sent
Eve's Presence	No	[No, Yes]	Affects ability to detect Eve (Affects gain and yield)
Signal loss in Bob (dB)	[3.5-7.8]	[7.5]	Affects gain and yield
Detector Efficiency (%)	[6.5-16.5]	[10]	Affects gain and yield
Detector Dark Count Probability	[1E-5-3E-5]	[2.5E-5]	Affects gain and yield
Detector After Pulse Probability	[0.001-0.0235]	[0.008]	Affects gain and yield

Factors to hold constant are listed in Table 15 are a large subset of the total number of experimental factors that are held constant. These factors are somewhat independent of the particular simulation selected and have a non-negligible impact to a QKD system. Only the Error Correction Method and Protocol are categorical factors; the rest are numerical. NOTE: The error rate threshold is used by the protocol after the error rate is calculated. If the calculated error rate is greater than the specified threshold, the assumption is that an eavesdropper is interfering during quantum exchange and the protocol will terminate at this point.

Table 15. Factors to Hold Constant

Variable/Factor	Desired Level or Allowable Range	Measurement Precision	Anticipated Effects on Responses or Objective
Error correction method	decoyPerfect	N/A	Simplified error rates
Error rate threshold	0.25	0.01	Eavesdropper detection ability
Maximum # of frames	12000	1	Actual simulation run time
Number of detections	50,000	1	Terminating condition for simulation
Protocol	BB84	N/A	High security
Pulses per frame	1000	1	Affects simulation duration
Signal loss in fiber (dB/km)	0.2	0.05	Lower yield over distance

Nuisance Factors

Due to the fact that experiment will be conducted via computer simulation, the nuisance factors are negligible. That is, many of the nuisance factors are controlled by holding them constant or excluding them from the simulation.

Known/Suspected Interactions

The specified percentages and MPN for signal, decoy, and vacuum states are expected to affect the efficiency of the yield over greater distances and thus the likelihood of detecting an eavesdropper during the quantum exchange phase of the BB84 protocol.

Restrictions

Due to the nature of computer simulations, the experiment has few restrictions. For example, the simulated phenomena are programmed to obey the laws of physics and nature. Again by using computer simulation, it is relatively easy to adjust the control variables.

Two practical realities affect the results of the experiment: data and time. First, the author anticipates that any data required will be available, but possibly not in the most convenient format. Furthermore, he anticipates there will be a learning curve in how to evaluate the data the simulation produces. Second, the other large restriction will be proportional to the amount of time require to execute the test runs. As the control variables suggest, there are a large number of factors and levels that could be evaluated. Thus the experiment may take a lot of time to flesh out the relevant experimental regions from the irrelevant.

Design Preferences

The author anticipates that replication will be the most effective design principle to employ. Replication will enable the analyst to build confidence intervals and make comparisons between the observations. Blocking, in the sense of batch processing may be employed to conduct test runs. Randomization can be used to plan the order in which to adjust the control variables. Again, due to the nature of computer simulation, some principles of blocking and randomization are irrelevant.

Analysis & Presentation techniques

Analysis will involve the use of box plots to perform a preliminary visual examination of the output data. If the data appears to be within the range of expectations, confidence intervals on the difference of the estimated signal and decoy state yields will be calculated. These confidence intervals will be represented in a tabular format or graphically as necessary.

Coordination

Weekly teleconferences are normally conducted with the entire team and attended by the sponsor and SMEs. Additionally, coordination was required between the AFIT QKD team and AFIT's Center for Cyberspace Research in order to obtain additional computer hardware for simulation processing.

Necessity of Trial Runs

Trials runs are a necessity. The plan is as follows:

1. Conduct runs to verify the author's understanding of the configured system and BB84 protocol.
2. Conduct additional runs to verify that control variables are adjusted correctly by comparing the observations to expectations.
3. Conduct additional trial runs as necessary.

Simulation Environment

The decoy state qkdX architecture will be configured and simulated on 1 HP Workstation which has 32 cores. Thus, with each core running a single simulation, 32 simulation runs can be executed simultaneously. This set of runs will be split across two virtual machines. Generally, each machine will run one treatment per core at a time performing the simulation using a different random number seed.

The decoy state enabled qkdX architecture was organized to minimize the number of locations in which simulation configuration modifications would be made. Table 16 outlines the locations of the control variables necessary to configure the notional architecture. Similarly, the file locations of the *Factors to Hold Constant* are identified in Table 17.

Table 16. Control Variable File Locations.

Simulation Variable	**qkdX File Location**
Signal State MPN	qkdX/examples/MainDse1/omnetpp.ini
Decoy State MPN	qkdX/examples/MainDse1/omnetpp.ini
% of signal states	qkdX/examples/MainDse1/omnetpp.ini
% of decoy states	qkdX/examples/MainDse1/omnetpp.ini
% of vacuum states	qkdX/examples/MainDse1/omnetpp.ini
Distance	qkdX/examples/MainDse1/omnetpp.ini
Quantum Channel Signal Loss (controlled by distance)	qkdX/examples/MainDse1/omnetpp.ini
Signal loss in Bob	qkdX/examples/MainDse1/omnetpp.ini
Eve's Presence	qkdX/examples/MainDse1/omnetpp.ini
Detector efficiency (%)	qkdX/examples/MainDse1/omnetpp.ini
Dark count probability (%)	qkdX/examples/MainDse1/omnetpp.ini
Detector After Pulse Probability	qkdX/examples/MainDse1/omnetpp.ini

Table 17. File Locations for Factors to Hold Constant.

Variable/Factor	qkdX Location
Error Correction Method	qkdX/examples/MainDse1/omnetpp.ini
Error rate threshold	qkdX/examples/MainDse1/omnetpp.ini
Maximum # of frames	qkdX/lib/sys/src/sys/config-sys.h
Number of detections	qkdX/examples/MainDse1/omnetpp.ini
% of vacuum states	qkdX/examples/MainDse1/omnetpp.ini
Protocol	qkdX/examples/MainDse1/omnetpp.ini
Pulses per frame	qkdX/lib/sys/src/sys/config-sys.h
Signal loss in fiber (dB/km)	qkdX/examples/MainDse1/omnetpp.ini

The maximum number of frames will be set after trial runs of the experiment. This value affects the amount of memory consumed by the simulation for various internal variables. It has already been observed that increases in the order of magnitude for this variable translate to additional processing time. Once the trials runs have been completed, this variable will be set to a constant value and will not vary with treatments.

Simulation Control Variables

The mean photon number is achieved by attenuating a classical laser pulse down to the desired mean number of photons. Hence both MPN and attenuation are listed, but only attenuation is directly set in the simulation.

Treatments

Another way to view the control variables is shown in Table 18. Here the control variables are separated into levels and one can easily see there are 2 combinations of interest in a full factorial design of this experiment. Thus one comparison can be made using this baseline.

Table 18. Control Variable Summary

Eve's Presence	Distance	Channel Loss	Detector Efficiency	Loss in Bob	Dark Count Probability (Rate)	After Pulse Probability	MPN Signal	MPN Decoy	Signal/Decoy/Vacuum Percentage
No	20km	5.6 dB	10%	7.5 dB	2.5e-5	0.008	0.65	0.08	75/12.5/12.5
Yes									

Appendix C. Decoy State Enabled QKD System Use Cases

Request Secret Key

Summary: A pair of human secret key users, e.g. human Alice (H.Alice) and human Bob (H.Bob), request a secret key from the Quantum Key Distribution (QKD) system. This use case initiates the process in which the QKD system grows a symmetric secret key for the secret key user pair.

Dependency: Extended by Perform BB84 and Perform Decoy State Enabled BB84

Actor(s): Human Secret Key Users, e.g., H.Alice and H.Bob

Resources: QKD System

Pre-conditions: QKD System is powered on. Each secret key user has the necessary QKD system components available to him/her. The QKD system is able to perform communication actions between components available to each user. The QKD system has a well-characterized quantum communications channel.

Post-condition: QKD System provided a symmetric secret key to the secret key users.

Main sequence:

1. H.Alice requests a secret key from the QKD system using the decoy state enabled BB84 protocol.
2. Perform Decoy State Enabled BB84.
3. The QKD system provides a symmetric secret key to H.Alice and H.Bob.

Alternative sequence:

1. Human secret key user Alice (H.Alice) requests a secret key from the QKD system using the standard BB84 protocol.
   1.1. Perform BB84.

Design Decisions:

Outstanding questions/issues:

Perform Decoy State Enabled BB84

Summary: QKD System components, Alice and Bob, grow a secret key by executing an implementation of a decoy state enabled BB84 protocol. This protocol enables the system to detect an eavesdropper performing a photon number splitting attack during the exchange of quantum bits.

Dependency:

```
       Extends Request Secret Key
       Includes:
            Authenticate Partners (Authentication)
            Exchange Quantum Bits Using Decoy Protocol (Quantum Exchange)
            Generate Sifted Key Using Decoy Protocol (Raw Key Sifting)
            Detect Eavesdropper
                  Estimate Signal QBER (Error Estimation)
                  Reconcile Signal Errors (Error Reconciliation)
            Estimate Entropy (Entropy Estimation)
            Perform Privacy Amplification (Privacy Amplification)
            Generate Secret Key (Key Generation)
```

Actor(s): Human Secret key users

Resources: QKD System and System of interest

Pre-condition: QKD System is powered on. Each secret key user has the necessary QKD system components available to him/her. The QKD system is able to perform communication actions between components available to each user. The systems have been primed with initial key (from manual keying or from a key store from a previous session). The QKD system has a well-characterized quantum communications channel (thresholds are known). The decoy state enabled BB84 protocol has been selected for use.

Post-condition: The quantum subsystem has grown a symmetric secret key that will be provided to the human secret key users.

Main sequence:

1. <<Authenticate Partners (Authentication)>>
2. <<Exchange Quantum Bits Using Decoy Protocol (Quantum Exchange)>>
3. <<Generate Sifted Key Using Decoy Protocol (Raw Key Sifting)>>
4. <<Detect Eavesdropper>>
   4.1. <<Estimate QBER Using Decoy Protocol (Error Estimation)>>
   4.2. <<Reconcile Errors Using Decoy Protocol (Error Reconciliation)>>
5. <<Estimate Entropy (Entropy Estimation)>>
6. <<Perform Privacy Amplification (Privacy Amplification)>>
7. <<Generate Secret Key (Key Generation)>>

Alternative sequence:

- TODO Error rate is too high
- TODO Secret key cannot be generated
- TODO An eavesdropper is detected while performing a PNS attack

Design Decisions:

Outstanding questions/issues:

Authenticate Partners (BB84 – Step 1: Authentication)

Summary: Alice and Bob are identified and proven to be who they claim to be. This is the first step in the BB84 protocol and is commonly known as *Authentication*. This step enables the secret key partners to be certain of each other's identity across the classic communications channel.

Dependency: Included by Standard BB84 and Decoy State Enabled BB84.

Actor(s): Secret key users (partners)

Resources: System of interest

Precondition: BB84 protocol has been initiated. QKD System is powered on. Each secret key user has the necessary QKD system components available to him/her. The QKD system is able to perform communication actions between components available to each user. Alice and Bob are using an information theoretically secure classical authentication algorithm.

Post-Condition: Partners (including subsystems) have been authenticated.

Main Sequence:

1. Establish communication
2. Initiate authentication.
3. Identify partners.
4. Authenticate identities.
5. Each partner's identity is authenticated.

Alternative Sequence(s):

Design Decisions:

- During the modeling process, this phase is assumed to occur successfully. It is not the primary behavior being studied.

Outstanding questions/issues:

Exchange Quantum Bits Using Decoy State Protocol (BB84 – Step 2: Quantum Exchange)

Summary: Alice and Bob prepare and measure pulses, using a polarization-encoding scheme and decoy states, until they have enough raw key to perform the next step of the BB84 protocol. This is the second step in the BB84 protocol and is commonly known as *Quantum Exchange*. This phase is where a PNS attack could occur and produces much of the data that will be used by the remaining phases.

Dependency: Included by Perform Decoy State Enabled BB84. Extended by <<Calibrate System>>

Actor(s): Alice Protocol (Alice) and Bob Protocol (Bob)

Resources: System of interest

Pre-condition: Partner's identities have been authenticated. Classical communication channel is established and authenticated. Calibration has occurred. Polarization and timing synchronization are assumed to remain constant.

Post-condition: Enough raw key exists to perform sifting and obtain the desired block size.

Table 19. Alice and Bob Knowledge at End of Exchange Quantum Bits Using Decoy State Protocol

Alice Knows:	Bob Knows:
- Prepared pulse bit values - Prepared pulse bases - Prepared pulse type - Pulse timing information - Total number of each type of pulse sent	- Measured pulse bit values - Measured pulse bases - Pulse timing information - Total number of detections

Main sequence:

1. Alice and Bob agree to start quantum exchange.
2. Alice and Bob set their frame counters to zero.
3. Alice and Bob establish quantum exchange and frame parameters.
   3.1. Timing pulse (shared information between Alice and Bob; also used for polarization control)
   3.2. Pulse duration
   3.3. Pulses per frame
   3.4. Frame to frame delay
   3.5. Block size (e.g. 10K signal states)
   3.6. Number of detections before sifting (Bob's detection limit)
   3.7. Signal, decoy, and vacuum state MPNs
   3.8. Signal, decoy, and vacuum state percentages
   3.9. Maximum number of frames before sifting must occur (related to Alice's memory limit)
4. Alice chooses a random sequence of polarization bases in which to encode photons.
5. Alice chooses a random bit string.

6. Alice randomly determines the state type (based on pre-configured percentages of signal, decoy, and vacuum) for each qubit in the random bit string and stores the decisions.
7. Alice stores the frame number, basis, value, and state type for each pulse in the frame.
8. Alice transmits a frame to Bob
   8.1. Alice transmits a timing pulse.
   8.2. Alice produces pulses and encodes qubits according to the bit values, basis sequence, and state type.
9. Alice increments the frame number.
10. Alice verifies that frame number is less than the maximum number of frames before sifting.
11. Alice waits for the frame to frame delay before sending the next frame.
12. Bob receives the frame from Alice.
    12.1. Bob receives the timing pulse.
    12.2. Bob receives the weak coherent pulses (WCPs).
13. For each WCP in the frame, Bob, randomly and independently from Alice, selects and stores a basis in which to measure the qubit.
14. Bob attempts to detect photons and measure the bit values for each WCP in the frame.
15. Bob stores the bit values from the detected photons using the bases he selected.
16. Beginning with step 3, this sequence repeats until the Bob's detections the number of photons before sifting.
17. Bob stops detecting pulses.
18. Bob stops recording bit values.
19. Bob records the timing information of the last pulse he detected.
20. Bob transmits a message requesting Alice to stop sending frames.
21. Alice receives Bob's message.
22. Alice finishes sending the frame.
23. Alice acknowledges Bob's request to stop and stops sending frames.
24. Alice and Bob agree to proceed to the next phase.

Alternative sequence:

8. The maximum number of frames has been reached (i.e., Alice runs out of memory)
9. Alice and Bob agree to proceed to the next phase (raw key sifting).

Design Decisions:

Outstanding questions/issues:

- To be implemented in the future: Alarm/error condition is set if no detections are achieved after 2 attempts

Generate Sifted Key Using Decoy State Protocol (BB84 – Step 3: Raw Key Sifting)

Summary: Alice and Bob compare the measurement bases and drop the bases that do not match. Alice calculates statistics related to quantum exchange. Alice sends the decoy bit values to Bob for use later. This is the third step in the BB84 protocol and is more commonly known as *Sifting*.

Dependency: Included by Perform Decoy State Enabled BB84.

Actor(s): Alice Protocol (Alice) and Bob Protocol (Bob)

Resources: System of interest

Pre-condition: Partner's identities have been authenticated. Classical communication channel is established and authenticated. Calibration has occurred. No error conditions have occurred.

Post-condition: Alice and Bob have raw secret keys of equal length and matching bases.

Table 20. Alice and Bob Knowledge at End of Generate Sifted Key Using Decoy State Protocol

Alice has/knows:	Bob has/knows:
- Which of her frames/slots for which Bob measured a matching basis - Number of signal, decoy, and vacuum detections - Number of pulses sent before Bob's detection limit was reached - Number of signal, decoy, and vacuum pulses sent before the detection limit was reached - Number of signal, decoy, and vacuum pulses sent - Dark count rate	- Which measured frames/slots matched the basis prepared by Alice - Which frames and slots contained a signal, decoy, or vacuum pulse - Number of signal, decoy, and vacuum detections - Dark count rate - Decoy bit values

Main sequence:

1. Alice and Bob exchange messages to initiate sifting.
2. Bob announces to Alice the measurement bases and the timing information, e.g. frame number and slot, he used for measuring qubits.
3. Alice receives the information from Bob.
4. Alice identifies and stores the signal detections where the basis matched.
5. Alice identifies and stores the decoy detections where the basis matched.
6. Alice discards the bits where the Bob's measurement basis did not match Alice's prepared basis.
7. Alice determines or calculates:
   7.1. Alice determines and counts the total number of signal, decoy, and vacuum detections.
   7.2. Alice calculates and stores the total number of pulses sent before the detection limit was reached.

- 7.3. Alice calculates and stores the total number of signal, decoy, and vacuum pulses sent before the detection limit was reached.
- 7.4. Alice calculates and stores the total number of signal, decoy, and vacuum pulses sent.

8. Alice calculates (but does not store) the signal, decoy, and rates.
9. Alice calculates and stores the vacuum detection (i.e., dark count) rate
10. Alice prepares a message to Bob that:
    - 10.1. identifies the bases he correctly matched,
    - 10.2. identifies which bits were signal and decoy,
    - 10.3. contains the dark count rate, and
    - 10.4. contains decoy bit values.
11. Bob receives the message from Alice.
12. Bob identifies and stores
    - 12.1. the signal bits with which his basis measurement matched Alice's and
    - 12.2. the decoy bits with which his basis measurement matched Alice's.
13. Bob stores:
    - 13.1. the decoy bit values from Alice and
    - 13.2. the dark count rate from Alice.
14. Alice and Bob exchange messages to terminate sifting.

Alternative sequence:

Design Decisions:

- Alice announces decoy bits to Bob in this phase. Though it may make sense for clarity to perform this action in another phase, we chose to implement it here to make mitigate use of the classic communications channel which is a large source of practical system delays.
- Alice provides Bob with the dark count rate. This was implemented here in an attempt to make the protocol more efficient.

Outstanding questions/issues:

- If Alice is the "master" in this process, why does she send the dark count to Bob?
    - I don't think she needs to do this.

Detect Eavesdropper

Summary: Alice and Bob calculate signal and decoy error rates, reconcile signal errors and attempt to detect an eavesdropper as well as detect a PNS attack. This step is unique to the decoy state enabled BB84 protocol and includes the *Error Estimation* and *Error Reconciliation* phases of BB84.

Dependency: Includes <<Estimate QBER (BB84 – Step 4: Error Estimation)>> and <<Reconciliation Signal Errors (BB84 – Step 5: Error Reconciliation)>>.

Actor(s): Alice Protocol (Alice) and Bob Protocol (Bob)

Resources: System of interest.

Pre-condition: Partner's identities have been authenticated. Classical communication channel is established and authenticated. Calibration has occurred. Pre-established tolerances for the signal and decoy photon number dependent yield differences and a pre-established threshold for the estimated and actual signal and decoy QBERs are known.

Post-condition: No eavesdropper has been detected performing a PNS attack and the protocol is ready to continue.

Table 21. Alice and Bob Knowledge at End of Detect Eavesdropper

Alice Knows:	Bob Knows:
- Actual signal and decoy QBERs - Estimated signal QBER - Signal and decoy gains - End to end efficiencies, η_μ,η_ν - 1-, 2-, 3-, 4-, 5-photon signal and decoy yield estimates - Whether or not tolerance was exceeded	- Actual signal and decoy QBERs

Main sequence:

1. <<*Estimate Signal QBER*>>.
2. <<*Reconcile Signal Errors*>>.
3. Alice checks whether the actual decoy QBER exceeds the actual decoy QBER threshold.
4. Alice calculates and stores the measured gain by dividing the total number of signal and decoy detections by the total number of signal and decoy pulses she sent to Bob.

$$Q_\mu = \sum_{n=0}^{\infty} \frac{e^{-\mu}\mu^n}{n!} Y_n = Y_0 + 1 - e^{-\mu\eta_\mu} = \frac{\#\, of\ signal\ detections}{\#\, of\ signal\ pulses\ sent^*}$$

Signal gain (use case) (22)

$$Q_\nu = \sum_{n=0}^{\infty} \frac{e^{-\nu}\mu^n}{n!} Y_n = Y_0 + 1 - e^{-\nu\eta_\nu} = \frac{\#\, of\ decoy\ detections}{\#\, of\ decoy\ pulses\ sent^*}$$

Decoy gain (use case)

(23)

5. Alice uses the measured signal and decoy gains to calculate and store the end to end efficiencies.

$$\eta_\mu = \frac{-\ln|1 + Y_0 - Q_\mu|}{\mu}$$

End-to-end efficiency (use case; signal) (24)

$$\eta_\nu = \frac{-\ln|1 + Y_0 - Q_\nu|}{\nu}$$

End-to-end efficiency (use case; decoy) (25)

6. Alice uses the efficiency to calculate and store the estimated 1-, 2-, 3-, 4-, and 5-photon yield estimates for the signal and decoy states. The calculations for single (1-) photon yield estimates, Y1, are shown below:

$$signal\ Y_1 \cong Y_0 + \left[1 - \left(1 - \eta_\mu\right)^1\right] = Y_0 + \left[1 - \left(1 - \frac{-\ln|1 + Y_0 - Q_\mu|}{\mu}\right)^1\right]$$

Single photon signal yield estimate (use case) (26)

$$decoy\ Y_1 \cong Y_0 + [1 - (1 - \eta_\nu)^1] = Y_0 + \left[1 - \left(1 - \frac{-\ln|1 + Y_0 - Q_\nu|}{\nu}\right)^1\right]$$

Single photon decoy yield estimate (use case) (27)

7. Alice compares the 1-, 2-, 3-, 4-, and 5-photon yield of the signal states to the 1-, 2-, 3-, 4-, and 5-photon yield decoy states and stores the results. The difference is within a pre-established tolerance. The calculation for single (1-) photon yield estimate difference is:

$$|signal\ Y_1 - decoy\ Y_1| \leq tolerance_1$$

Single photon yield difference comparison to tolerance (use case) (28)

8. Alice does not detect a photon number splitting attack.

Alternative sequence:

7. Alice compares the single photon yield of the signal to the single photon yield decoy states. The difference is not within a pre-established tolerance.
8. Alice detects a photon number splitting attack, records the deviation and displays a warning.

Design Decisions:

- Definitions [14]:
    - Yield = the conditional probability of a detection event at Bob's side given that Alice sends out a pulse
    - Gain = product of the probability of Alice sending out a pulse (following a Poisson distribution) and the conditional

probability that Alice’s pulse (and background) will cause a detection at Bob

- Integrated Decoy State Error Test into this use case because it became a trivial (one line) use case.
- * That is, the number of pulses sent in the time frame during which Bob is ready to detect photons.
- For simulation and certain analytical studies it is better to warn a user of a difference exceeding a tolerance rather than terminating the simulation. Strict termination would prevent data capture.

Outstanding questions/issues:

- NOTE: End to end efficiency is related to, protocol (e.g., 80% of signal states), channel (0.2 dB/km), Bob’s internal loss (2-5dB), and detector efficiency (10%) and dark counts.
- When 2-photon, 3-photon, 4-photon, or n-photon yields are desired for comparison, the tolerance and difference for the n-photon yield must be calculated. For example:

$$signal\ Y_n \cong Y_0 + \left[1-\left(1-\eta_\mu\right)^n\right] = Y_0 + \left[1-\left(1-\frac{-\ln\left|1+Y_0-Q_\mu\right|}{\mu}\right)^n\right]$$

Photon number dependent signal yield estimate (use case) (29)

$$\left|signal\ Y_n - decoy\ Y_n\right| \leq tolerance_n$$

Photon number dependent yield difference comparison to tolerance (use case) (30)

- The idea of tolerance is consistent with the notion that Alice and Bob must know their channel well [28].
- How could calibrated end-to-end efficiencies be employed in Step 5?

Estimate Signal Quantum Bit Error Rate (BB84 – Step 4: Error Estimation)

Summary: Alice and Bob estimate the number of signal bit errors or quantum bit error rate (QBER) that occurred during quantum exchange. If the number of errors exceeds a pre-determined threshold (percentage), they decide to terminate the protocol. This is the fourth step in the BB84 protocol and is commonly known as *Error Estimation*. This step provides a quick test to detect the presence of an eavesdropper and provides error reconciliation methods with the error estimate.

Dependency:

Actor(s): Alice Protocol (Alice) and Bob Protocol (Bob)

Resources: System of interest

Pre-condition: Partner's identities have been authenticated. Classical communication channel is established and authenticated. Calibration has occurred. Alice and Bob have raw sifted keys. A predetermined percentage of signal bits (to be used for error estimation) is known by Alice. An estimated signal QBER threshold is known by Bob.

Post-condition: Alice and Bob have calculated a QBER estimate.

Table 22. Alice and Bob Knowledge at End of Estimate Signal QBER

Alice Knows:	Bob Knows:
- Which signal bits were selected for error estimation - Which signal bits were not selected for error estimation - Bob's estimated signal QBER	- Actual bit values for error estimation bits - Estimated signal QBER

Main sequence:

1. Alice and Bob exchange messages to initiate error estimation.
2. Alice uses a predetermined percentage to randomly select a sequence of sifted signal bits. These bits will be used for estimating the number of errors that occurred during the quantum exchange.
3. Alice separates the random sequence of bits from the sifted signal bits. The remaining sifted bits are identified as the post error estimation sifted signal bits.
4. Alice tallies the number of post error estimation sifted signal bits.
5. Alice transmits this sequence of bit values to Bob on the classic channel.
6. Bob receives the sequence of bits values from Alice.
7. Bob compares each bit value from Alice to each bit value he measured.
8. Bob calculates the percentage of errors measured in this comparison. This value is the estimated signal quantum bit error rate (QBER).
9. Bob checks whether the estimated signal QBER exceeds the estimated signal QBER threshold.
10. Bob announces the estimated signal QBER to Alice on the public channel.
11. Alice receives the message from Bob.

12. Alice stores the estimated signal QBER.
13. Alice checks whether the estimated signal QBER exceeds the estimated signal QBER threshold.
14. Alice and Bob terminate error estimation.

Alternative sequence:

9. If the QBER estimate exceeds a threshold, Alice and Bob terminate the protocol.
13. If the signal QBER estimate exceeds a threshold, Alice and Bob terminate the protocol.

Design decisions:

- This use case is shared between the decoy state enabled BB84 and standard BB84. The estimated signal error rate is a parameter to some error reconciliation methods. Additionally it provides an opportunity to test for an eavesdropper on the quantum channel by comparing the estimated signal QBER to a predefined threshold.
- In a meeting at the research sponsor’s location during the week of 17 November 2014, it was established that using the decoy states for error estimation is unnecessary for (at least) two reasons. The error reconciliation methods that depend on the estimated signal QBER; an estimated decoy state QBER, which uses a different MPN is not an appropriate substitute for the estimated signal QBER. Additionally, the decoy state error count can be calculated in another phase or use cases. This practice would be consistent with the desired software engineering practices of loose coupling and high cohesion. Thus no decoy states are used in this phase.

Outstanding questions/issues:

Reconcile Signal Errors (BB84 – Step 5: Error Reconciliation)

Summary: Alice and Bob compare qubit measurements and reconcile discrepancies between using a two-way error correction method. Bob also calculates the quantum bit error rate (QBER). This is the fifth step in the BB84 protocol and is commonly known as *Error Reconciliation*.

Dependency: Extended by *Execute Perfect ER*, *Execute LDPC ER*, *Execute Cascade ER*, or *Execute Winnow ER*.

Actor(s): Alice Protocol (Alice) and Bob Protocol (Bob)

Resources: System of interest

Pre-condition: Partner’s identities have been authenticated. Classical communication channel is established and authenticated. Calibration has occurred. Alice and Bob estimated the number of errors that occurred during quantum exchange. Alice and Bob agreed upon a particular error reconciliation method. A predetermined actual signal QBER threshold has been defined and known by Bob.

Post-condition: Alice and Bob have reconciled the erroneous bits in their sifted keys.

Table 23. Alice and Bob Knowledge at End of Reconcile Signal Errors

Alice knows:	Bob knows:
- Error reconciled signal bit values	- Error reconciled signal bit values

Main sequence:

1. Alice and Bob exchange messages to initiate this phase.
2. Alice and Bob <<*Execute [Error Reconciliation Name]*>> method.
3. Alice and Bob terminate this phase.

Alternative sequence:

Design Decisions:

- The selected Error Reconciliation method should perform any QBER-related checks. This decision was made to keep the protocol efficient. Refer to Execute Decoy Perfect ER as an example.
- The bulk of this use case exists in the specific error reconciliation method.

Outstanding questions/issues:

Execute Decoy Perfect ER

Summary: Alice provides Bob with the bit values she selected for the secret key. This is a simple method of error reconciliation that is used for modeling purposes only.

Dependency: Extends Signal Reconcile Errors

Actor(s): Alice Protocol (Alice) and Bob Protocol (Bob)

Resources: System of interest

Pre-condition: Partner's identities have been authenticated. Classical communication channel is established and authenticated. Calibration has occurred. Alice and Bob agreed to use the Decoy Perfect Error Reconciliation method.

Post-condition: Alice and Bob have reconciled the erroneous bits in their sifted signal and decoy keys.

Table 24. Alice and Bob Knowledge at End of Execute Decoy Perfect ER

Alice Knows:	Bob Knows:
- Actual signal and decoy QBERs	- Actual signal and decoy bit values - Actual signal and decoy QBERs

Main sequence:

1. Alice and Bob exchange messages to initiate this phase.
2. Alice updates her reconciled key using the values from the post error estimated sifted signal bits.
3. Alice transmits the values of the signal and decoy bits to Bob on the classical communication channel.
4. Bob receives the bits values from Alice.
5. Bob adjusts his measured signal bit values to match the bit values he received from Alice.
6. Bob counts the number of mismatched bits, i.e. actual number of signal bit errors and actual number of decoy bit errors.
7. Bob stores the actual number of signal bit errors.
8. Bob calculates the actual signal and decoy QBERs.
9. Bob checks whether the actual signal QBER exceeds the actual signal QBER threshold.
10. Bob announces the actual signal and decoy QBERs to Alice.
11. Alice receives the actual signal and decoy QBERs from Bob.
12. Alice stores actual signal and decoy QBERs.
13. Alice and Bob terminate this phase.

Alternative sequence:

9. If the QBER does exceed the actual signal QBER threshold, Alice and Bob terminate the protocol.

Design Decisions:

- The calculation of the actual signal QBER may not be explicitly part of any error reconciliation method. Thus Alice or Bob must calculate it. Using the theoretical perfect error reconciliation method, in which Alice sends all of the signal bit values to Bob across the classical channel, it makes the most sense for Bob to calculate the actual signal QBER. Alternatively, an additional sequence of messages transfers on the classic communication channel would be performed at the time the QBER would need to be calculated.
- Bob announces the number of signal and decoy errors to Alice. Alice has been assumed to be the leader in the protocol. Thus it makes sense for her to track and store the information.

Outstanding questions/issues:

- Bob probably doesn’t need to store the actual number of signal bit errors.
- Alice probably doesn’t need to send the decoy bit values during sifting.

Execute Decoy State Error Test (Integrated into Detect Eavesdropper Use Case)

Summary: Alice provides Bob with the decoy bit values she selected for the decoy states key.

Dependency: Extends Reconcile Errors

Actor(s): Alice Protocol (Alice) and Bob Protocol (Bob)

Resources: System of interest

Pre-condition: Partner's identities have been authenticated. Classical communication channel is established and authenticated. Calibration has occurred. An Error Reconciliation method has been performed. Alice knows the actual decoy QBER.

Post-condition: The actual decoy QBER is at or below a predefined actual decoy QBER threshold.

Main sequence:

1. Alice initiates this phase.
2. Alice checks whether the actual decoy QBER exceeds the actual decoy QBER threshold.
3. Alice terminates this phase.

Alternative sequence:

2. If the actual decoy QBER exceeds the actual decoy QBER threshold, Alice and Bob terminate the protocol.

Design Decisions:

- **After analyzing this use case, it was integrated into the Detect Eavesdropper use case because the value added by this use was a single check performed by main sequence step #2.**

- The bulk of this use case is handled via the precondition that Alice knows the value of the actual decoy QBER. The decoy perfect ER method was designed to calculate the decoy QBER as a matter of efficiency since Bob already had all of the information needed. Alternatively, an unnecessary sequence of messages transfers on the classic communication channel would be performed at this time.

Outstanding questions/issues:

Appendix D. Other BB84 Use Cases

Perform BB84

Summary: QKD System components, Alice and Bob, grow a secret key by executing the BB84 protocol.

Dependency:

Extends Request Secret Key
Includes:
Authenticate Partners (Authentication)
Exchange Quantum Bits (Quantum Exchange)
Generate Sifted Key (Raw Key Sifting)
Estimate Signal Quantum Bit Error Rate (Error Estimation)
Reconcile Signal Errors (Error Reconciliation)
Estimate Entropy (Entropy Estimation)
Perform Privacy Amplification (Privacy Amplification)
Generate Secret Key (Key Generation)

Actor(s): Human Secret key users

Resources: QKD System and System of interest

Pre-condition: QKD System is powered on. Each secret key user has the necessary QKD system components available to him/her. The QKD system is able to perform communication actions between components available to each user. The QKD system has a well-characterized quantum communications channel. The (standard) BB84 protocol has been selected for use.

Post-condition: The quantum subsystem has grown a symmetric secret key that will be provided to the human secret key users.

Main sequence:

1. <<*Authenticate Partners (Authentication)* >>
2. <<*Exchange Quantum Bits (Quantum Exchange)* >>
3. <<*Generate Sifted Key (Raw Key Sifting)* >>
4. <<*Estimate Signal Quantum Bit Error Rate (Error Estimation)*>>
5. <<*Reconcile Signal Errors (Error Reconciliation*)>>
6. <<*Estimate Entropy (Entropy Estimation)*>>
7. <<*Perform Privacy Amplification (Privacy Amplification)*>>
8. <<*Generate Secret Key (Key Generation)* >>

Alternative sequence:

Design Decisions:

Outstanding questions/issues:

Exchange Quantum Bits (BB84 – Step 2: Quantum Exchange)

Summary: Alice and Bob prepare and measure pulses, using a polarization-encoding scheme, until they have enough raw key to perform the next step of the BB84 protocol. This is the second step in the BB84 protocol and is commonly known as *Quantum Exchange*.

Dependency: Included by *Perform BB84 Protocol*. Includes *Calibrate System*.

Actor(s): Alice Protocol and Bob Protocol

Resources: System of interest

Pre-condition: Partners have been authenticated. Classical communication channel is established.

Post-condition: Enough raw key exists to perform sifting. Quantum systems are calibrated.

Main sequence:

1. *<<Calibrate System>>*
2. Alice and Bob establish quantum exchange and frame parameters.
   2.1. Timing pulse
   2.2. Pulse duration
   2.3. Pulses per frame
   2.4. Block size (e.g. 10K signal states)
   2.5. Maximum number of detections (detection limit)
3. Alice chooses a random sequence of polarization bases in which to encode photons and stores the sequence.
4. Alice chooses a random bit string and stores it.
5. All state types are signal states.
6. Alice produces pulses and encodes qubits according to the bit values, basis sequence, state type, and frame parameters.
7. Alice transmits the frame to Bob.
8. Bob receives the frame from Alice.
9. For each pulse in the frame, Bob, randomly and independently from Alice, selects and stores a basis in which to measure the qubit.
10. Bob attempts to detect photons and measure the bit values for each pulse in the frame.
11. Bob stores the bit values from the detected photons using the bases he selected.
12. Steps 1-10 repeat until the two participants decide enough bits have been detected to proceed.

Alternative sequence:

Design Decisions:

Outstanding questions/issues:

- Describe the relationship between Alice and Bob.
- Frames are used to facilitate timing

Generate Sifted Key (BB84 – Step 3: Raw Key Sifting)

Summary: Alice and Bob compare the measurement bases and drop the bases that do not match. This is the third step in the BB84 protocol and is commonly known as *Sifting*.

Dependency: Included by *Perform BB84*.

Actor(s): Secret key users, Alice and Bob

Resources: Quantum Subsystem

Pre-condition: Partners have been authenticated. Classical communication channel is established. A predetermined number of quantum pulses were exchanged between Alice and Bob, i.e. detection limit was met.

Post-condition: Alice and Bob have raw secret keys of equal length and matching bases.

Main sequence:

1. Bob announces to Alice the measurement bases and the timing information, e.g. frame number and slot, he used for measuring qubits.
2. Alice compares Bob's bases to the ones she prepared and transmitted.
3. Alice discards the bits for which Bob did not select the correct basis.
4. Alice announces to Bob the bases he correctly matched.
5. Bob discards the bits for which he did not select the correct basis.
6. If decoy states are enabled, Bob stores the state types announced by Alice.

Alternative sequence:

Design Decisions:

Outstanding questions/issues:

Estimate Entropy (BB84 – Step 6: Entropy Estimation)

Summary: Alice and Bob estimate the amount of information that was exposed to potential eavesdroppers on the classical and quantum channels. This is the sixth step in the BB84 protocol and is commonly known as *Entropy Estimation*.

Dependency:

Actor(s): Secret key users

Resources: Quantum Subsystem

Pre-condition: The QBER has been calculated.

Post-condition: Entropy has been estimated.

Main sequence:

1. TODO in future research

Alternative sequence:

Design Decisions:

Outstanding questions/issues:

- This use case should be defined in future research which focuses on this set of behaviors.

Perform Privacy Amplification (BB84 – Step 7: Privacy Amplification)

Summary: Alice and Bob attempt to mitigate the impact of the amount of information that was exposed to potential eavesdroppers. This is the seventh step in the BB84 protocol and is commonly known as *Privacy Amplification*.

Dependency:

Actor(s): Secret key users

Resources: Quantum Subsystem

Pre-condition:

Post-condition:

Main sequence:

1. TODO in future research

Alternative sequence:

Design Decisions:

Outstanding questions/issues:

- This use case should be defined in future research which focuses on this set of behaviors.

Generate Secret Key (BB84 – Step 8: Key Generation)

Summary: Alice and Bob verify they have matching secret keys and distribute the keys for use with the bulk encryptors. This is the last (eighth) step in the BB84 protocol and is commonly known as *Key Generation*.

Dependency:

Actor(s): Secret key users

Resources: Quantum Subsystem

Pre-condition:

Post-condition:

Main sequence:

1. TODO in future research

Alternative sequence:

Design Decisions:

Outstanding questions/issues:

- This use case should be defined in future research which focuses on this set of behaviors.

Appendix E. Initial Requirements

Table 25. Initial Requirements Notation.

Requirement Types:	Verification methods:
OR – operational requirement	I – Inspection
UR – user-specified requirement	D – Demonstration
DR – derived requirement	T – Testing
SS – system specification	A – Analysis

1. The model shall employ the Bennett and Brassard (BB84) protocol (OR, D).
    1.1. The model shall transfer encoded bits on a “quantum” channel (DR, I).
    1.2. The model shall encode bit values on quantum particles (OR, I).
    1.3. The model shall decode bit values from quantum particles (OR/DR, I).
2. The model shall extend the BB84 protocol with the decoy state protocol (OR, I).
    2.1. The decoy state protocol shall use three states: signal, decoy, and vacuum (DR, I).
3. The model shall detect an eavesdropper performing a photon number splitting attack on the quantum channel (OR, T or A).
    3.1. The model shall use the decoy state protocol to detect a PNS attack (DR, I).
4. The model shall use commercially-available components when available (UR, I).

Appendix F: Requirements

Table 26. Requirements Notation.

Requirement Types:	Verification methods:
OR – operational requirement UR – user-specified requirement DR – derived requirement SS – system specification	I – Inspection D – Demonstration T – Testing A – Analysis

Table 27. Requirements.

Requirement Number	**Definition**	**Requirement Type**	**Primary Verification Method**
1	**The model shall employ the Bennett and Brassard (BB84) protocol.**	OR	D
1.1	**The model shall transfer encoded bits on a "quantum" channel.**	DR	I
1.1.1	**The model shall adhere to a predefined timing scheme.**	DR	T
1.1.1.1	The model shall synchronize timing between Alice and Bob so that weak coherent pulses (WCPs) can be detected every 1.5 nanoseconds.	DR	D
1.1.1.2	**Alice shall generate laser pulses according to a predefined timing scheme.**	DR	T
1.1.1.2.1	The timing scheme shall be defined by an optical pulse frame which consists of a timing pulse and at least one weak coherent pulse.	DR	I
1.1.1.2.2	Alice shall transmit optical pulse frames on the quantum channel.		D
1.1.1.2.3	Alice shall be capable of generating laser pulses with a 400 picosecond duration at least every 1.5 nanoseconds	DR	I
1.1.1.2.4	Alice shall indicate the beginning of a pulse frame by transmitting a timing pulse with an MPN of at least 6,000,000.	DR	I
1.1.1.2.5	The timing pulse shall have a duration of at least 400 picoseconds and less than 600 picoseconds.	DR	D
1.1.1.2.6	The number of weak coherent pulses in a pulse frame shall be configurable between 1 and at least 10,000.	DR	D
1.1.1.2.7	Weak coherent pulses shall have a duration of 400 picoseconds and less than 600 picoseconds.	DR	D
1.1.1.2.8	Alice shall transmit optical pulse frames until Bob announces he has detected a predefined number of qubits or Alice reaches her memory limit.	DR	D
1.1.1.3	Bob shall announce to Alice when he has detected a predefined number of qubits.	DR	D
1.1.1.4	Bob shall be ready to detect pulses according to a predefined timing scheme.	DR	T
1.1.2	**Alice shall encode bit values on quantum particles.**	OR	I

Requirement Number	Definition	Requirement Type	Primary Verification Method
1.1.2.1	Alice shall use photon polarization to encode bit values onto quantum particles.	UR, OR	I
1.1.2.2	Alice shall randomly select either rectilinear or diagonal polarization as the qubit encoding basis.	DR	I
1.1.2.3	Alice shall store the bases she selects to prepare qubits.	DR	I
1.1.2.4	Alice shall randomly select bit values to encode onto quantum particles	DR	I
1.1.2.5	Alice shall store the bit values she selects to prepare qubits.	DR	I
1.1.3	The model shall decode bit values from quantum particles.	OR	I
1.2	**Alice and Bob shall perform sifting.**	OR	D
1.2.1	Bob shall announce to Alice when he detected each photon.	DR	D
1.2.2	Bob shall announce to Alice the sequence of bases he selected to measure qubits.	DR	D
1.2.3	Bob shall announce to Alice when he measured a qubit detection.	DR	D
1.2.4	Alice shall announce to Bob the sequence of bases she selected to encode qubits.	DR	D
1.2.5	Alice shall discard qubits for which her prepared bases do not match Bob's measured bases.	DR	D
1.2.6	Bob shall discard qubits for which his measured bases do not match Alice's prepared bases.	DR	D
1.2.7	Alice shall calculate and store the dark count rate.	DR	I
1.3	Alice and Bob shall reconcile errors in their respective sifted keys.	DR	D
2	**The model shall extend the BB84 protocol with the decoy state protocol.**	DR	I, D
2.1	**The decoy state protocol shall use three states: signal, decoy, and vacuum**	DR	I
2.1.1	**Alice shall encode signal states using the signal state MPN specification that ranges between 0.00 and 15.00.**	DR	I
2.1.1.1	Alice shall transmit signal states according to a prescribed occurrence percentage that ranges between 0 and 100%.	DR	I
2.1.1.2	Alice shall store the number of signal states she transmitted.	DR	I
2.1.2	**Alice shall encode decoy states using the decoy state MPN specification that ranges between 0.00 and 15.00.**	DR	I
2.1.2.1	Alice shall randomly transmit decoy states according to a prescribed occurrence percentage that ranges between 0 and 100%.	DR	I
2.1.2.2	Alice shall store the number of decoy states she transmitted.	DR	I
2.1.3	**Alice shall encode vacuum states using the vacuum state MPN specification of 0.00**	DR	I
2.1.3.1	Alice shall randomly transmit vacuum states according to a prescribed occurrence percentage that ranges between 0 and 100%.	DR	I

Requirement Number	Definition	Requirement Type	Primary Verification Method
2.1.3.2	Alice shall store the number of vacuum states she transmitted.	DR	I
2.1.4	Alice shall announce to Bob the sequence of state types she transmitted.	DR	I
2.1.5	Alice shall calculate and store the gain of the signal state qubits.	DR	D
2.1.6	Alice shall calculate and store the gain of the decoy state qubits.	DR	D
2.1.7	Alice shall calculate and store the end-to-end signal efficiency	DR	D
2.1.8	Alice shall calculate and store the end-to-end decoy efficiency	DR	D
2.1.9	Alice shall calculate and store the yield of the 1-photon signal pulses, i.e., the 1-photon signal yield.	DR	D
2.1.10	Alice shall calculate and store the yield of the 2-photon signal pulses, i.e., the 2-photon signal yield.	DR	D
2.1.11	Alice shall calculate and store the yield of the 3-photon signal pulses, i.e., the 3-photon signal yield.	DR	D
2.1.12	Alice shall calculate and store the yield of the 4-photon signal pulses, i.e., the 4-photon signal yield.	DR	D
2.1.13	Alice shall calculate and store the yield of the 5-photon signal pulses, i.e., the 5-photon signal yield.	DR	D
2.1.14	Alice shall calculate and store the yield of the 1-photon decoy pulses, i.e., the 1-photon decoy yield.	DR	D
2.1.15	Alice shall calculate and store the yield of the 2-photon decoy pulses, i.e., the 2-photon decoy yield.	DR	D
2.1.16	Alice shall calculate and store the yield of the 3-photon decoy pulses, i.e., the 3-photon decoy yield.	DR	D
2.1.17	Alice shall calculate and store the yield of the 4-photon decoy pulses, i.e., the 4-photon decoy yield.	DR	D
2.1.18	Alice shall calculate and store the yield of the 5-photon decoy pulses, i.e., the 5-photon decoy yield.	DR	D
2.1.19	Alice shall calculate and store the difference of the 1-photon signal yield and the 1-photon decoy yield.	DR	D
2.1.20	Alice shall calculate and store the difference of the 2-photon signal yield and the 2-photon decoy yield.	DR	D
2.1.21	Alice shall calculate and store the difference of the 3-photon signal yield and the 3-photon decoy yield.	DR	D
2.1.22	Alice shall calculate and store the difference of the 4-photon signal yield and the 4-photon decoy yield.	DR	D
2.1.23	Alice shall calculate and store the difference of the 5-photon signal yield and the 5-photon decoy yield.	DR	D
3	**The model shall detect an eavesdropper performing a photon number splitting attack on the quantum channel.**	OR	T or A
3.1	**The model shall use the decoy state protocol to detect a PNS attack.**	DR	I

Requirement Number	**Definition**	**Requirement Type**	**Primary Verification Method**
3.1.1	The model shall detect an eavesdropper performing a photon number splitting (PNS) attack on the quantum channel by verifying that the difference of the 1-photon signal and decoy yield estimates is less than 0.00202 photons/pulse sent.	DR	T
3.1.2	The model shall detect an eavesdropper performing a photon number splitting (PNS) attack on the quantum channel by verifying that the difference of the 2-photon signal and decoy yield estimates is less than 0.00399 photons/pulse sent.	DR	T
3.1.3	The model shall detect an eavesdropper performing a photon number splitting (PNS) attack on the quantum channel by verifying that the difference of the 3-photon signal and decoy yield estimates is less than 0.00592 photons/pulse sent.	DR	T
3.1.4	The model shall detect an eavesdropper performing a photon number splitting (PNS) attack on the quantum channel by verifying that the difference of the 4-photon signal and decoy yield estimates is less than 0.00780 photons/pulse sent	DR	T
3.1.5	The model shall detect an eavesdropper performing a photon number splitting (PNS) attack on the quantum channel by verifying that the difference of the 5-photon signal and decoy yield estimates is less than 0.00965 photons/pulse sent	DR	T
4	**The model shall imitate behavior of commercially-available components when available.**	UR	I
4.1	**The model shall use one or more lasers to produce signal, decoy, and vacuum state pulses.**	DR	I
4.1.1	The laser(s) shall produce pulses with wavelengths in its operating range between 1290 nm and 1600 nm.	DR	I
4.1.2	The laser(s) shall be capable of producing pulses at a specified wavelength within the operating range +/-5nm.	DR	I
4.1.3	Bob shall correct for polarization drift with 99% accuracy.	DR	D
4.1.4	**Bob shall detect at least one in ten weak coherent pulses which are 400 picoseconds in duration.**	DR	D
4.1.4.1	Bob shall be capable of detecting a weak coherent pulse every 1.5 nanoseconds.	DR	D
4.1.4.2	The detector dark count rate shall be configurable to interfere with Bob's ability to detect a weak coherent pulse.	DR	D
4.1.4.3	The detector after pulse rate shall be configurable to interfere with Bob's ability to detect a weak coherent pulse.	DR	D
5	**The QKD model shall be accurate, flexible, usable, extensible and low cost.**	UR	D
5.1	**The QKD model shall be able to be reconfigured without re-compiling the code for the following parameters.**	DR	D
5.1.1	The mean photon number of the signal states shall be configurable between 0.000 and 15.000 without recompiling the	DR	D

Requirement Number	Definition	Requirement Type	Primary Verification Method
	simulation.		
5.1.2	The mean photon number of the decoy states shall be configurable between 0.000 and 15.000 without recompiling the simulation.	DR	D
5.1.3	The mean photon number of the vacuum states shall be configurable between 0.0 and 0.01.	OR	D
5.1.4	The percentage of signal states shall be configurable between 0.00 and 100.00 without recompiling the simulation.	OR	D
5.1.5	The percentage of decoy states shall be configurable between 0.00 and 100.00 without recompiling the simulation.	OR	D
5.1.6	The percentage of vacuum states shall be configurable between 0.00 and 100.00 without recompiling the simulation.	OR	D
5.1.7	The percentage of bits used for error estimation shall be configurable between 0.00 and 100.00 without recompiling the simulation.	DR	D
5.1.8	The percentage of estimated errors that result in restarting the protocol shall be configurable between 0.00 and 100.00 without recompiling the simulation.	DR	D
5.1.9	The number of laser pulses per second shall be configurable between 1,000,000 and 1,000,000,000 without recompiling the simulation.	DR	D
5.1.10	The length of the quantum channel, in meters, shall be configurable with 0.001 precision without recompiling the simulation.	DR	D
5.1.11	The amount of attenuation in quantum channel, in optical dB per kilometer, shall be configurable with 0.001 precision without recompiling the simulation.	DR	D
5.1.12	The dark count rate, in percent, shall be configurable with 0.000 000 001 precision without recompiling the simulation.	DR	D
5.1.13	The after pulse rate, in percent, shall be configurable with 0.000 000 001 precision without recompiling the simulation.	DR	D
5.1.14	The random number generator seed shall be configurable between 1 and 100,000,000 without recompiling the simulation.	DR	D
5.2	The QKD model shall produce output in OMNeT++ scalar (.sca) format for MPN (Signal).	DR	D
5.3	The QKD model shall produce output in OMNeT++ scalar (.sca) format for MPN (Decoy).	DR	D
5.4	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of pulses sent from Alice to Bob.	DR	D
5.5	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of signal pulses sent from Alice to Bob.	DR	D
5.6	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of decoy pulses sent from Alice to Bob.	DR	D

Requirement Number	Definition	Requirement Type	Primary Verification Method
5.7	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of vacuum pulses sent from Alice to Bob.	DR	D
5.8	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of signal pulses detected by Bob.	DR	D
5.9	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of decoy pulses detected by Bob.	DR	D
5.10	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of vacuum pulses detected by Bob.	DR	D
5.11	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of sifted signal pulses.	DR	D
5.12	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Total number of sifted decoy pulses.	DR	D
5.13	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Gain (signal).	DR	D
5.14	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Gain (decoy).	DR	D
5.15	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Dark count rate.	DR	D
5.16	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Signal efficiency.	DR	D
5.17	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Decoy efficiency.	DR	D
5.18	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (1-photon signal yield).	DR	D
5.19	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (2-photon signal yield).	DR	D
5.20	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (3-photon signal yield).	DR	D
5.21	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (4-photon signal yield).	DR	D
5.22	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (5-photon signal yield).	DR	D
5.23	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (1-photon decoy yield).	DR	D
5.24	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (2-photon decoy yield).	DR	D
5.25	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (3-photon decoy yield).	DR	D
5.26	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (4-photon decoy yield).	DR	D

Requirement Number	Definition	Requirement Type	Primary Verification Method
5.27	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Yield estimate (5-photon decoy yield).	DR	D
5.28	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Number of signal errors.	DR	D
5.29	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Number of decoy errors.	DR	D
5.30	The QKD model shall produce output in OMNeT++ scalar (.sca) format for Number of error-reconciled key bits.	DR	D
6	**The QKD model shall be able to run on at least one high performance computer available to the AFIT QKD research team.**	UR	D

Bibliography

[1] DocForge, "DocForge - Software Development Resources (Architecture)," [Online]. Available: http://docforge.com/wiki/Architecture. [Accessed 11 12 2014].

[2] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, 1984.

[3] V. J. Easton and J. H. McColl, "Statistics Glossary," [Online]. Available: http://www.stats.gla.ac.uk/steps/glossary/index.html. [Accessed 16 08 2014].

[4] F. L. Ramsey and D. W. Schafter, The Statistical Sleuth, Boston: Brooks/Cole, 2013.

[5] DocForge, "DocForge - Software Development Resources (Framework)," [Online]. Available: http://docforge.com/wiki/Framework. [Accessed 11 12 2014].

[6] Y. Zhao, B. Qi, X. Ma, H. K. Lo and L. Qian, "Experimental quantum key distribution with decoy states," *Physical Review Letters,* vol. 96, no. 7, 2006.

[7] The Technical Cooperation Program, "USD-ATL: GUIDEx," 02 2006. [Online]. Available: http://www.acq.osd.mil/ttcp/guidance/documents/GUIDExBookFeb2006.pdf. [Accessed 11 12 2014].

[8] Air Force Institute of Technology, *Quantum Key Distribution Simulation Framework,* 29 October 2014 ed., M. R. Grimaila and L. O. Mailloux, Eds., Wright-Patterson AFB, Ohio, 2014.

[9] OMNeT++, "OMNeT++ User Manual," 12 2014. [Online]. Available: http://www.omnetpp.org/doc/omnetpp/manual/usman.htm. [Accessed 12 01 2015].

[10] M. R. Grimaila, J. D. Morris and D. D. Hodson, "Quantum Key Distribution," *The Information System Security Association (ISSA) Journal,* vol. 10, no. 6, pp. 20-27, June 2012.

[11] The Technical Cooperation Program, "TTCP GUIDEx Pocketbook," March 2006. [Online]. Available: http://www.acq.osd.mil/ttcp/guidance/documents/GUIDExPocketbookMar2006.pdf. [Accessed 17 01 2015].

[12] H. Gomaa, Software Modeling & Design, Cambridge: Cambridge University Press, 2011.

[13] D. C. Montgomery, Design and Analysis of Experiments, Danvers: John Wileys & Sons, Inc, 2013.

[14] X. Ma, B. Qi, Y. Zhao and H.-K. Lo, "Practical decoy state for quantum key distribution," *Physical Review A,* vol. 72, no. 1, p. 012326, 2005.

[15] L. O. Mailloux, R. D. Engle and M. R. Grimaila, "Modeling Decoy State Quantum Key Distribution Systems," *Journal of Defense Modeling & Simulation,* Submitted December 2014.

[16] J. Banks, J. S. Carson, B. L. Nelson and D. M. Nicol, Discrete Event System Simulation, 5, Ed., Upper Saddle River: Prentice Hall, 2010.

[17] D. D. Hodson, M. R. Grimaila and L. O. Mailloux, "Quantum Communications Simluation Toolbox, 1. Introduction_v2.0," 2013.

[18] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus and M. Peev, "The security of practical quantum key distribution," *Reviews of Modern Physics,* vol. 81, no. 3, pp. 1301-1350, 2009.

[19] B. Huttner, N. Imoto, N. Gisin and T. Mor, "Quantum cryptography with coherent states," *Physical Review A,* vol. 51, no. 3, pp. 1863-1869, 1995.

[20] G. Brassard, N. Lutkenhaus, T. Mor and B. C. Sanders, "Limitations on Practical Quantum Cryptography," *Physical Review Letters,* vol. 85, no. 6, pp. 1330-1333, 2000.

[21] N. Lutkenhaus, "Security against individual attacks for realistic quantum key distribution," *Physical Review A,* vol. 61, no. 052304, pp. 052304-1-052304-10, 2000.

[22] N. Lutkenhaus and M. Jahma, "Quantum key distribution with realistic states: photon-number statistics in the photon-number splitting attack," *New Journal of Physics,* vol. 4, pp. 44.1-44.9, 2002.

[23] V. Scarani, A. Acin, G. Ribordy and N. Gisin, "Quantum Cryptography Protocols Robust against Photon Number Splitting Attacks for Weak Laser Pulse Implementations," *Physical Review Letters,* vol. 92, no. 5, pp. 057901-1-057901-4, 2004.

[24] W.-Y. Hwang, "Quantum key distribution with high loss: Toward global secure communication," *Physical Review Letters,* vol. 91, no. 5, p. 057901, 2003.

[25] B. S. Blanchard and W. J. Fabrycky, Systems Engineering and Analysis, Boston: Prentice Hall, 2011.

[26] N. Gisin, G. Ribordy, W. Tittel and H. Zbinden, "Quantum cryptography," *Reviews of Modern Physics,* vol. 74, no. 1, pp. 145-195, 2002.

[27] DoD Deputy Chief Information Officer, "Background - DODAF," [Online]. Available: http://dodcio.defense.gov/TodayinCIO/DoDArchitectureFramework/dodaf20_background.aspx. [Accessed 11 12 2014].

[28] H.-K. Lo, X. Ma and K. Chen, "Decoy state quantum key distribution," *Physical Review Letters,* vol. 94, no. 3, p. 230504, 2005.

[29] X.-B. Wang, "Beating the photon-number-splitting attack in practical quantum cryptography," *Physical Review Letters,* vol. 94, no. 23, p. 230503, 2005.

[30] G. S. Vernam, "Cipher printing telegraph systems: For secret wire and radio telegraph communications," *AIEE, Journal of the ,* vol. 45, no. 2, pp. 109-115, 1926.

[31] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal,* vol. 28, no. 4, pp. 656-715, 1949.

[32] S. Singh, The Code Book, London: Fourth Estate, 2000.

[33] SE Handbook Working Group International Council on Systems Engineering, "Systems Engineering Handbook," INCOSE, San Diego, 2011.

[34] Defense Acquisition University, "Glossary of Defense Acquisition Acronyms and Terms," December 2012. [Online]. Available: https://dap.dau.mil/glossary/pages/2739.aspx. [Accessed 10 12 2014].

[35] INCOSE, "What is Systems Engineering?," INCOSE, 2014. [Online]. Available: http://www.incose.org/practice/whatissystemseng.aspx. [Accessed 03 12 2014].

[36] Defense Acquisition University, "[Systems Engineering] Introduction," Defense Acquisition University, 2014. [Online]. Available: https://acc.dau.mil/CommunityBrowser.aspx?id=638297. [Accessed 03 12 2014].

[37] Defense Acquisition University, "Defense Acquisition Guidebook: Chapter 4 - Systems Engineering," 2014. [Online]. Available: https://acc.dau.mil/CommunityBrowser.aspx?id=638295. [Accessed 11 12 2014].

[38] MITRE Corporate Communications and Public Affairs, MITRE Systems Engineering Guide, Bedford: The MITRE Corporation, 2014.

[39] International Organization for Standardization, "ISO/IEC 15288-2008: Systems and software engineering -- Systems life cycle processes," International Organization for Standardization, Geneva, 2008.

[40] Defense Acquisition University, "DAG Ch 5.1 Life-Cycle Sustainment in the Defense Acquisition Management System," DAU, [Online]. Available: https://acc.dau.mil/CommunityBrowser.aspx?id=489746. [Accessed 12 01 2015].

[41] Defense Acquisition University, "Systems Engineering Process," 2014. [Online]. Available: https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=9c591ad6-8f69-49dd-a61d-4096e7b3086c. [Accessed 11 12 2014].

[42] A. M. Law and D. W. Kelton, Simluation Modeling and Analysis, Third ed., Boston: McGraw Hil, 2000.

[43] R. G. Sargent, "Verification and Validation of Simluation Models," in *2010 Winter Simulation Conference*, Baltimore, 2010.

[44] O. Balci, "Verification, Validation, and Certification of Modeling and Simulation Applications," in *2003 Winter Simulation Conference*, New Orleans, 2003.

[45] Defense Acquisition University, "TST102: Fundamentals of Test and Evaluation," Defense Acquisition University, [Online]. Available: https://learn.dau.mil/Atlas2/html/scorm12/course/course_index.jsp?user_id=null&course_id=803352&course_prefix=TST&version=9&scorm_version=3&roster_id=-0.20423484770075795_803352&course_name=Fundamentals%20of%20Test%20and%20Evaluation&course_number=102&mod. [Accessed 14 01 2015].

[46] J. O. Grady, System Verification, Burlington: Academic Press, 2007.

[47] J. M. Colombi, *SENG 640 Systems Architecture - Lecture 9 - Process/Activity Modeling,* Dayton, OH, 2014.

[48] W. W. Royce, "Royce, W. W. (1970, August). Managing the development of large software systems," *IEEE WESCON,* vol. 26, no. 8, 1970.

[49] D. Nicolette, "Dave Nicolette: Effective Software Development and Delivery," 02 2012. [Online]. Available: https://davenicolette.files.wordpress.com/2012/02/rup.png. [Accessed 30 01 2015].

[50] A. L. Ramos, J. V. Ferreira and J. Barcelo, "Model-based system engineering: an emerging approach for modern systems," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 42, no. 1, pp. 101-111, 2012.

[51] International Council on Systems Engineering, "Systems Engineering Vision 2025," 23 June 2014. [Online]. Available: http://www.incose.org/newsevents/announcements/docs/INCOSE_SE_Vision_2025.pdf. [Accessed 20 February 2015].

[52] D. Long and Z. Scott, A Primer for Model-Based Systems Engineering, Vitech Corporation, 2011.

[53] J. A. Estefan, "INCOSE Survey of MBSE Methodologies," INCOSE, Seattle, 2008.

[54] S. Friedenthal, A. Moore and R. Steiner, A Practical Guide to SysML, Waltham: Elsevier, 2012.

[55] R. Cloutier, B. Sauser, M. Bone and A. Taylor, "Transitioning Systems Thinking to Model-Based Systems Engineering: Systemigrams to SysML Models," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS,* p. Accepted for publication, 2015.

[56] J. Holt, S. Perry, J. Bryans, Hallerstede and F. O. Handsen, "A Model-Based Approach for Requirements Engineering for Systems of Systems," *IEEE Systems Journal,* Accepted for publication.

[57] S. C. Spangelo, D. Kaslow, C. Delp, B. Cole, L. Anderson, E. Fosse, B. S. Gilbert, L. Hartman, T. Kahn and J. Cutler, "Applying model based systems engineering (mbse) to a standard cubesat," *In Aerospace Conference, 2012 IEEE,* pp. 1-20, 2012.

[58] A. Soyler and S. Sala-Diakanda, "A Model-Based Systems Engineering Approach to Capturing Disaster Management Systems," *In Systems Conference, 2010 4th Annual IEEE,* pp. 283-287, 2010.

[59] C/S2ESC - Software & Systems Engineering Standards Committee, "1362-1998 - IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document," IEEE Computer Society, 1998.

[60] Defense Acquisition University, "Concept of Operations," 2015. [Online]. Available: https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=f8f70a76-5dda-4346-9bfb-7ef0608f71bb.

[61] MITRE, "Concept of Operations," 2015. [Online]. Available: http://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/concept-development/concept-of-operations.

[62] Institute of Electrical and Electronics Engineers, "IEEE Standards Definition Database Search - M," [Online]. Available: http://dictionary.ieee.org/index/m-9.html. [Accessed 10 12 2014].

[63] H. D. Mills, "The management of software engineering, Part I: Principles of software engineering," *IBM Systems Journal ,* vol. 19, no. 4, pp. 414-420, 1980.

[64] N. Matloff, The Art of R Programming, San Francisco: No Starch Press, 2011.

[65] DoD Deputy Chief Information Officer, *DoD Architecture Framework Version 2.02,* 2010.

[66] S. W. Ambler, "Web services programming tips and tricks: Documenting a use case," IBM, 05 October 2000. [Online]. Available: http://www.ibm.com/developerworks/library/ws-tip-docusecase/. [Accessed 14 01 2015].

[67] G. Schneider and J. P. Winters, "Chapter 7: Documenting Use Cases," [Online]. Available: http://www.ibm.com/developerworks/rational/library/content/legacy/parttwo/1000/0670/0670_Schneider_Ch07.pdf. [Accessed 14 01 2015].

[68] T. Weilkiens, Systems Engineering with SysML/UML Modeling, Analysis, Design, Boston: The MK/OMG Press, 2006.

[69] Object Management Group (OMG), "Uniified Modeling Language (UML) Resource Page," OMG, 30 10 2014. [Online]. Available: http://www.uml.org/. [Accessed 15 02 2015].

[70] J. Colombi, "AFIT SENG 640 - Lecture 9: Process/Activity Modeling," Dayton, 2014.

[71] Eclipse Community, "Eclipse," 2015. [Online]. Available: https://www.eclipse.org/home/index.php. [Accessed 11 02 2015].

[72] OMNeT++ Community, "OMNeT++," OMNeT++ Community, [Online]. Available: http://omnetpp.org. [Accessed 11 02 2015].

[73] R. L. Rivest, "Cryptology," 1990. [Online]. Available: http://people.csail.mit.edu/rivest/Rivest-Cryptography.pdf. [Accessed 04 03 2015].

[74] B. Schneier, Applied Cryptography Second Edition: protocols, algorithms, and source code in C, New York: John Wiley & Sons, Inc, 1996.

[75] S. Wiesner, "Conjugate coding," *ACM Sigact News,* vol. 15, no. 1, pp. 78-88, 1983.

[76] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, 1984.

[77] D. Gottesman, H.-K. Lo, N. Lutkenhaus and J. Preskill, "Security of quantum key distribution with imperfect devices," in *In Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, 2004.

[78] V. Scarani and C. Kurtsiefer, "The black paper of quantum cryptography: real implementation problems," *arXiv:0906.4547v2,* 2009.

[79] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail and J. Smolin, "Experimental quantum cryptography," *Journal of Cryptology,* vol. 5, no. 1, pp. 3-28, 1992.

[80] R. Renner, N. Gisin and B. Kraus, "An information-theoretic security proof for QKD protocols," *arXiv:quant-ph/0502064v1,* 2005.

[81] L. O. Mailloux, M. R. Grimaila, D. D. Hodson, G. Baumgartner and C. McLaughlin, "Performance evaluations of quantum key distribution system architectures," *IEEE Security and Privacy,* vol. 15, no. 1, pp. 30-40, 2015.

[82] X.-B. Wang, "Decoy-state protocol for quantum cryptography with four different intensities of coherent light," *Physical Review A,* vol. 72, no. 1, p. 012322, 2005.

[83] Y. Zhao, B. Qi, X. Ma, H.-K. Lo and L. Qian, "Simulation and implementation of decoy state quantum key distribution over 60km telecom fiber," in *2006 IEEE International Symposium on Information Theory*, Seattle, 2006.

[84] X.-B. Wang, C.-Z. Peng and J.-W. Pan, "Simple protocol for secure decoy-state quantum key distribution with a loosely controlled source," *Applied Physics Letters,* vol. 90, no. 3, p. 031110, 2007.

[85] X.-B. Wang, L. Yang, C.-Z. Peng and J.-W. Pan, "Decoy-state quantum key distribution with both source errors and statistical fluctuations," *New Journal of Physics,* vol. 11, no. 7, p. 075006, 2009.

[86] J.-Z. Hu and X.-B. Wang, "Reexamination of the decoy-state quantum key distribution with an unstable source," *Physical Review A,* vol. 82, no. 1, p. 012331, 2010.

[87] J.-Z. Hu and X.-B. Wang, "Secure quantum key distribution in an easy way," *arXiv,* vol. arXiv:1004.3730, 2010.

[88] J. W. Harrington, J. M. Ettinger, R. J. Hughes and J. E. Nordholt, "Enhancing practical security of quantum key distribution with a few decoy states," *arXiv,* Vols. quant-ph/0503002, 2005.

[89] T. Horikiri and T. Kobayashi, "Decoy state quantum key distribution with a photon number resolved heralded single photon source," *Physical Review A,* vol. 73, no. 3, 2006.

[90] M. Hayashi, "General theory for decoy-state quantum key distribution with an arbitrary number of intensities," *New Journal of Physics,* vol. 9, no. 8, 2007.

[91] W. Mauerer and C. Silberhorn, "Quantum key distribution with passive decoy state selection," *Physical Review A,* vol. 75, no. 5, p. 050305, 2007.

[92] D. Rosenberg, J. W. Harrington, P. R. Rice, P. A. Hiskett, C. G. Peterson, R. J. Hughes and J. E. Nordholt, "Long-distance decoy-state quantum key distribution in optical fiber," *Physical Review Letters,* vol. 98, no. 1, 2007.

[93] W.-Y. Hwang, "Quantum key distribution with high loss: Toward global secure communication," *Physical Review Letters ,* vol. 91, no. 5, p. 057901, 2003.

[94] T. Y. Chen, H. Liang, Y. Liu, W. Q. Cai, L. Ju, W. Y. Liu and J. W. Pan, "Field test of a practical secure communication network with decoy-state quantum cryptography.," *Optics Express,* vol. 17, no. 8, pp. 6540-6549, 2009.

[95] L. O. Mailloux, J. D. Morris, M. R. Grimaila, D. D. Hodson, D. Jacques, J. M. Colombi, C. McLaughlin and J. A. Holes, "A Modeling Framework for Studying Quantum Key Distribution System Implementation Non-Idealities," *IEEE Access,* Submitted 29 Dec 2014.

[96] L. O. Mailloux, M. R. Grimaila, D. D. Hodson and C. McLaughlin, "Modeling Continuous Time Optical Pulses in a Quantum Key Distribution Discrete Event Simulation," in *International Conference on Security and Management SAM'14*, 2014.

[97] DoD Deputy Chief Information Officer, "Operational Viewpoint (OV-5a & OV-5b)," [Online]. Available: http://dodcio.defense.gov/TodayinCIO/DoDArchitectureFramework/dodaf20_ov5ab.aspx. [Accessed 13 01 2015].

[98] J. D. Morris, *Ph.D. Specialty Exam,* Wright-Patterson Air Force Base, 2012.

[99] INCOSE, "8th Annual INCOSE Great Lakes Regional Conference," 2014. [Online]. Available: http://www.incose.org/newsevents/events/details.aspx?id=252. [Accessed 11 02 2015].

[100] J. Zhang, R. Thew, C. Barreiro and H. Zbinden, "Practical fast gate rate InGaAs/InP single-photon avalanche photodiodes," *Applied Physics Letters,* p. 091103, 2009.

[101] INCOSE, "INCOSE Systems Engineering Vision 2020," Technical Operations INCOSE, 2007.

[102] C. Z. Peng, J. Zhang, D. Yang, W. B. Gao, H. X. Ma, H. Yin and J. W. Pan, "Experimental long-distance decoy-state quantum key distribution based on polarization encoding," *Physical Review Letters,* vol. 98, no. 1, 2007.

[103] The Fiber Optic Association, Inc., "The FOA Reference for Fiber Optics - Understanding dB," The Fiber Optic Association, Inc., [Online]. Available: http://www.thefoa.org/tech/ref/basic/dB.html. [Accessed 15 01 2015].

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD–MM–YYYY) 26-03-2015	2. REPORT TYPE Master's Thesis	3. DATES COVERED *(From — To)* Aug 2013 - Mar 2015

4. TITLE AND SUBTITLE
Modeling, Simulation, and Analysis of a Decoy State Enabled Quantum Key Distribution System

5a. CONTRACT NUMBER

5b. GRANT NUMBER
5713400-301-6448

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Engle, Ryan D L, Capt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENV-MS-15-M-181

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Laboratory for Telecommunication Sciences
Dr. Gerry Baumgartner
8080 Greenmead Drive
College Park MD 20740
gbaumgartner@ltsnet.net

10. SPONSOR/MONITOR'S ACRONYM(S)
LTS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
Quantum Key Distribution (QKD) is an emerging technology which uses the principles of quantum mechanics to provide unconditionally secure key distribution. QKD systems are unique in their ability to detect an eavesdropper's presence and are being marketed for applications where high levels of secrecy are required such as banking, government, and military environments. QKD systems are composed of electrical, optical, and electro-optical components. Their design requires expertise across multiple disciplines including computer science, computer engineering, electrical engineering, information theory, optical physics, and quantum physics. This multi-disciplinary nature makes QKD an ideal candidate for study using Model Based Systems Engineering (MBSE) Processes, Methods, and Tools (PMTs).
The primary research goal is to gain understanding of the operation and performance of the QKD decoy state protocol through the use of MBSE PMTs. The main research contributions include development of a decoy state model, validation of the this protocol in a QKD system model implementation, and confirmation that application of MBSE PMTs are critical to the understanding and analysis of complex systems. This work presents the first known application of MBSE PMTs to analyze a QKD system and provides utility to system developers, designers and analysts who seek to quantify performance and security.

15. SUBJECT TERMS
Quantum Key Distribution, Model-Based Systems Engineering, Modeling & Simulation, Decoy State Protocol, Photon Number Splitting Attack

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Dr. Michael R. Grimaila, AFIT/ENV
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	UU	200	19b. TELEPHONE NUMBER (Include Area Code) (937) 255-3636 x4800; Michael.Grimaila@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18