

*Nikolaos K. Papanikolaou*  
*Computer Systems Engineering*

---

**Formal Specification  
and Verification of  
Quantum Cryptographic  
Protocols**

---

**Third Year Project Report**

Supervisor: *Dr. Rajagopal Nagarajan*

DEPARTMENT OF COMPUTER SCIENCE  
AND SCHOOL OF ENGINEERING  
UNIVERSITY OF WARWICK  
COVENTRY

2003



## Abstract

THE SUBJECT of this thesis is a research project concerned with quantum cryptography, undertaken by the author in his final year of undergraduate study. The project deals with the application of computer modelling software to the protocols used for Quantum Key Distribution (QKD). Specifically, the goals of this project are twofold:

- (1) To *define* the behaviour of protocols for QKD formally (Specification), and
- (2) To *verify*, for a particular number of transmitted bits, that QKD is indeed secure against eavesdropping (Verification).

The approach taken to achieve these goals involves the use of formal methods. On one hand, a mathematical notation for the formal description of such protocols is developed. On the other hand, the CWB-NC and PRISM software tools are used to determine whether the BB84 protocol is secure against eavesdropping attacks.

The mathematical notation, named QKD-FORM, is merely a convenient formalism for listing the steps taken in popular QKD protocols. BB84, B92 and EPR are all expressed in QKD-FORM. This formalism is to be extended in future work so as to provide the basis for a computer language; this would allow quantum protocols to be expressed at a high level. Furthermore, such a language could potentially be translated to a modelling language such as CCS or PRISM.

Previous work by R. Nagarajan involving the use of CCS to model and check the security of BB84 is revisited. The existing CCS models are explained in detail and shown diagrammatically. The greater part of the discussion focuses on the use of the PRISM software tool as a vehicle of investigation. With PRISM, we have been able to practically verify whether eavesdropping can be detected in BB84 for a specific number of bits; in addition, the exact probabilities of this occurring for key lengths of 1 to 64 bits are computed. The probability of detecting eavesdropping increases with the number of bits. This is indeed what the relevant theory predicts. The result is not new, since physicists have written several proofs of the security of BB84; however, the computer modelling approach is novel, and is capable of describing certain properties of actual implementations.



## **Key Words and Phrases**

- quantum cryptography,
- protocol design and validation,
- probabilistic modelling,
- formal specification,
- formal verification,
- CCS,
- PRISM.



## Acknowledgements

Supplementary advice was required concerning the nature and workings of the PRISM tool and this advice was sought and obtained via a couple of visits to the University of Birmingham (where the tool was developed). The author is particularly grateful to:

- **Dr. David Parker,**
- **Dr. Gethin Norman,** postdoctoral students at the University of Birmingham, and
- their former supervisor, **Prof. Marta Kwiatkowska.**

Their help in modelling protocols in the PRISM language and interpreting the program's output is highly appreciated.

I would also like to take the opportunity to thank my supervisor, **Dr. Rajagopal Nagarajan,** without whose support this project would not have been possible.

**My parents** have been a source of endless encouragement and moral support during what has been a very trying year for me. I can never thank them enough.



*This work is dedicated to my parents.*



## Author's Assessment of the Project

**T**HE AUTHOR has enjoyed undertaking this project considerably, despite its subject matter being quite advanced for an undergraduate student. The project has achieved its original aims and also:

- It is one of the first undergraduate projects dealing with Quantum Information Theory undertaken in a Computer Science department;
- It has *successfully* applied tried and tested methods of Computer Science (formal methods) to the novel field of quantum cryptography;
- It has provided the community with a formal notation, QKD-FORM for listing the steps in quantum cryptographic protocols; and
- It has taken an entirely new approach (computer-aided probabilistic verification) to proving the security of the BB84 protocol.

Furthermore, the project has given him a sufficient level of knowledge to further this work as a postgraduate student. The following are *personal achievements* of the author:

- He had the opportunity to give two related talks in the Warwick Department of Computer Science:
  - An introductory talk on Quantum Cryptography was given on November 7th, 2002 to fourth-year MEng students in the context of the CS406 *Research Directions in Computing* module.
  - An Open Day talk on Quantum Computation was given on March 5th, 2003 to college students.
- He was able to apply and extend my undergraduate knowledge of the concurrency formalism CCS.
- He obtained experience in using a specialist software tool and collaborated with its developers.
- He was able to apply my knowledge of C and UNIX shell scripting languages.

Evidently, this project has been very rewarding and a great challenge.

N. P.



## Preface

*Humans are incapable of securely storing high quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.)*

— C. KAUFMAN, R. PERLMAN, M. SPECINER,  
Network Security – Private Communication in a Public World

THE CAMBRIDGE International Dictionary of English defines a *protocol* as a ‘system of rules and acceptable behaviour used at official ceremonies and occasions’. Indeed, the concept of a protocol is used in many fields, particularly where effective communication is necessary. International diplomacy, for example, involves the establishment of protocols or conventions, which are formal agreements between various nations. The term ‘protocol’ has entered the domain of Computer Science fairly recently, in order to define the procedures used for the exchange of data within computer networks.

The need for formal communication procedures has been recognised since ancient times, as they provide a means of exchanging information in a manner as unambiguous as possible. The Greek historian Polybius (Πολύβιος) discussed the problems that arose when fire signals were used to communicate the fall of Troy to Athens, in Book X, Chapter 43 of his *Historiae* (Ἱστορίαι). Clearly, fire signals are a very limited means of communication. They were used quite commonly, however, at the time (2 B.C.). The extract in Figure 0.1 presents Polybius’ views on fire signals, and is considered one of the first ever descriptions of signalling methods known to us today (see also [1]):

Polybius’ account goes on to describe the limitations of fire signals, and the improvements made by Aeneas (Αἰνεΐας). The historian makes the remarkable observation that the greatest problem of all is how to communicate unexpected events (see Figure 0.2).

Polybius’ remark is equally valid to the present day. Because all communication systems have to be designed in order to account for unexpected events and potential errors.

Consequently, great care must be taken in designing communication protocols; the procedures have to be able to cope with the unexpected. In engineering terms, all possibilities have to be predicted by the protocol designer, from low-level issues (such as hardware problems) to high-level issues (such as the interoperability of the communicating systems).

<p>Τοῦ δὲ κατὰ τὰς πυρσεῖας γένους, μεγίστας δὴ παρεχομένου χρεΐας ἐν τοῖς πολεμικοῖς, ἀνεργάστου πρότερον ὑπάρχοντος, χρήσιμον εἶναι μοι δοκεῖ τὸ μὴ παραδραμεῖν, ἀλλὰ ποιήσασθαι περὶ αὐτοῦ τὴν ἀρμόζουσαν μνήμην. ὅτι μὲν οὖν ὁ καιρὸς ἐν πᾶσι μεγάλην ἔχει μερίδα πρὸς τὰς ἐπιβολάς, μεγίστην δ' ἐν τοῖς πολεμικοῖς, παντὶ δῆλον. τῶν δὲ πρὸς τοῦτο συναγωνισμάτων πλείστην ἔχουσι δύναμιν οἱ πυρσοί. ἄρτι γὰρ τὰ μὲν γέγονε, τινὰ δ' ἀκμήν ἐνεργεῖται καὶ δυνατὸν ἐστὶ γινώσκειν, ὧς μέλει, ποτὲ μὲν ἡμερῶν τριῶν ἢ τεττάρων ὁδὸν ἀπέηοντι, ποτὲ δὲ καὶ πλειόνων. ὥστ' αἰεὶ τοῖς δεομένοις πράγμασιν ἐπικουρίας παραδόξον γίνεσθαι τὴν βοήθειαν διὰ τῆς τῶν πυρσῶν ἀπαγγελίας.</p>	<p>The method of signalling by fire, which is of the highest utility in the operations of war, has never before been clearly expounded; and I think I shall be doing a service if I do not pass it over, but give an account of it adequate to its importance. Now that opportuneness is of the utmost moment in all undertakings, and pre-eminently so in those of war, no one doubts; and of all the things which contribute to enable us to hit the proper time nothing is more efficacious than fire signals. For they convey intelligence sometimes of what has just happened, sometimes of what is actually going on; and by paying proper attention to them one can get this information at three or four days' journey off, and even more: so that it continually happens that the help required may be unexpectedly given, thanks to a message conveyed by fire signals.</p>
<p>(From <i>The Histories of Polybius</i>, translation of F. Hultsch's text by Evelyn S. Shuckburgh.)</p>	

FIGURE 0.1. *Polybius, Historiae, Book X, Chapter 43*

<p>... ὅτι δὲ μεταβάλλονταί τινες τῶν πολιτῶν ἢ προδιδόασιν, ἢ φόνος ἐν τῇ πόλει γέγονεν, ἢ τι τῶν τοιούτων, ἃ δὴ συμβαίνει μὲν πολλάκις, <b>πρόληψιν δ' ἔχειν πάντων ἀδύνατον, —μάλιστα τὰ δὲ παραδόξως γινόμενα τῆς ἐκ τοῦ καιροῦ συμβουλίας καὶ ἐπικουρίας προσδεῖται—</b> τὰ τοιαῦτα πάντα διέφυγε τὴν τῶν πυρσῶν χρεῖαν. περὶ ὧν γὰρ οὐκ ἐνεδέχετο προνοηθῆναι, περὶ τούτων οὐδὲ σύνθημα ποιήσασθαι δυνατὸν.</p>	<p>... it was impossible to express that “certain citizens had gone over to the enemy,” or “were betraying the town,” or that “a massacre had taken place,” or any of those things which often occur, <b>but which cannot all be anticipated. Yet it is precisely the unexpected occurrences which demand instant consideration and succour.</b> All such things were naturally beyond the competence of fire signalling, inasmuch as it was impossible to adopt an arbitrary sign for things which it was impossible to anticipate.</p>
<p>(From <i>The Histories of Polybius</i>, translation of F. Hultsch's text by Evelyn S. Shuckburgh.)</p>	

FIGURE 0.2. *Polybius, Historiae, Book X, Chapter 44*

Computer scientists and engineers have tried to tackle these issues in various ways. It is generally preferable to try to eliminate all possibilities at the design stage, before even considering how a protocol is to be implemented.

It was in the 1970's and 1980's, when the communications industry was booming, that protocol design was studied with particular fervour. The protocols used in many modern networks (including the Internet) were designed carefully at the time. Despite numerous efforts to make these as efficient and error-free as possible, these networks are still far from perfect.

One of the more successful approaches to protocol design was proposed by Gerard Holzmann and put forward in his out-of-print volume *The Design and Validation of Computer Protocols* [1]. Holzmann suggested the use of formal methods for describing and checking the behaviour of communication protocols. In particular, he developed a protocol validation language, PROMELA, “in order to be able to study [protocol] structure and to verify their completeness and logical consistency” [1] (p. 90). The use of formal methods for specifying and verifying system behaviour is not new; but the application of such methods to protocols is of utmost importance.

The significance of these methods becomes even more apparent when protocols have to achieve *secure* communication. A certain class of protocols, known as *cryptographic protocols* or *security protocols*, deserves special attention from this point of view.

Cryptographic protocols are used in several areas, ranging from financial and industrial communications to governmental applications. Since these kinds of communication are often exposed to security threats, cryptographic protocols have to be thoroughly tested and approved before being used.

It is noteworthy that a major flaw in a well-known security protocol, the *Needham–Schroeder Public Key protocol*, was detected and corrected using formal methods [2]. Since this discovery, techniques for formally specifying and verifying cryptographic protocols have gained considerable recognition.

In the meantime, a novel, special kind of cryptographic protocol has emerged: *quantum cryptographic protocols*. These protocols use the principles of quantum mechanics in order to definitively solve the classical problem of key distribution. It was Stephen Wiesner that first made the connection between quantum physics and security, in his classical paper *Conjugate Coding* [4]. Charles Bennett and Gilles Brassard developed Wiesner’s ideas further, resulting in the so-called BB84 protocol. This was the genesis of quantum cryptography. Several similar protocols have been proposed since.

It is in the interest of computer scientists to begin dealing with quantum cryptographic protocols seriously. At present, there already exist implementations of these protocols. Moreover, the experimental physics associated with quantum cryptography is coming of age. Notably, BB84 has been shown to be unconditionally secure against *all* possible attacks.

It is reasonable to ask: is history repeating? It seems so. The very issues which Polybius identified with fire signals will be issues with quantum protocols. There will be, of course, much more of “the unexpected”. Quantum cryptographic protocols have an ingredient that classical protocols do not: *randomness*. Any useful attempt at describing and checking the behaviour of quantum protocols must incorporate this notion.

The question at the heart of this document is: can we apply the ideas of formal specification and validation to quantum protocols? and if so, can we use today’s computers to model their behaviour? It will be shown that, not only are such ideas relevant, but necessary to ensure the integrity of the quantum cryptographic systems of today and tomorrow.

## Outline

This document is divided into three parts.

The first part, *Preliminaries*, develops all the background material relevant to the specification and verification of quantum cryptographic protocols. In particular:

- **Chapter 1** presents the basics of protocols and quantum cryptography. It serves the purpose of a literature survey and formally defines the concepts assumed in subsequent chapters.
- **Chapter 2** details the three most important protocols (BB84, B92 and entanglement-based QKD i.e. the EPR protocol) that have been proposed to implement quantum key distribution. Of these three protocols, BB84 is the one used in our analysis.

The second part of the report, *Modelling and Analysis of the BB84 Protocol*, focuses on the techniques used to model BB84 and the results obtained using the CWB-NC and PRISM software:

- **Chapter 3** revisits previous work using CCS and the CWB-NC;
- **Chapter 4** presents a probabilistic model of BB84 in PRISM. It is here that the main results about BB84 are shown.

The final part of the report is devoted to a presentation of a formal notation used to describe BB84, B92 and EPR.

- **Chapter 5** defines the syntax and semantics of the formal notation QKD-FORM, and
- **Chapter 6** concludes by pointing directions for future work and summarizing the results.

## Additional Comments

By its nature, this project cannot account for all of the latest developments. However, an attempt has been made to reference all the relevant resources and latest literature.

The project has run quite smoothly since the beginning of the year, however several changes have been made to the original plan. The most important of these is the adoption of the PRISM modelling tool as the main vehicle of investigation. Originally the intention had been to experiment with the CWB-NC tool and the specification language CCS. This tool has been used but to a lesser extent, since the alternative is much more powerful.

Time management was supported by the Microsoft Project software, and a special website at <http://www.dcs.warwick.ac.uk/~esvbb/> was established to incorporate all project-related material.

The online preprint archive hosted at <http://uk.arXiv.org/> was of great use in this project, and references to articles of the archive are given in the form `quant-ph/*****`, where `*****` is a number that uniquely identifies the article.

A large part of the first term was devoted to surveying the relevant literature and obtaining copies of necessary academic papers. The resultant references list is quite lengthy, so each chapter of the report has a separate list; the citations in each chapter refer only to *that* chapter's bibliography.

## Nomenclature and Conventions

The notation used in this document is as follows:

- The bold letters **A, B, E** denote “Alice”, “Bob” and “Eve” respectively, taken conventionally to be the names of the two legitimate communicating parties and the eavesdropper.
- Other capital letters denote sequences:
  - Capital letters such as  $B$  and  $D$  denote sequences of values belonging to Alice.
  - Primed capital letters such as  $B'$  and  $D'$  denote sequences of values belonging to Bob.
  - Calligraphic  $\mathcal{M}$  is used for quantum measurement operators.
  - Calligraphic  $\mathcal{P}$  is used to denote probabilities.
- Sequences of values are expanded using either a set-like notation, e.g.

$$\{\beta_i \mid 0 \leq i \leq n-1\}$$

or equivalently using an enumeration, e.g.

$$[\beta_0, \beta_1, \dots, \beta_{n-1}]$$

- Cryptographic keys are written as sequences, but are named using two- or three- letter acronyms, such as RCK for “raw cryptographic key”, or TFK for “tentative final key”.

N. P.

Warwickshire, 2002

## Preface Bibliography

- [1] G. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.
- [2] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.
- [3] G. Simmons. Cryptanalysis and protocol failures. *Communications of the ACM*, 37(11), 1994.
- [4] S. Wiesner. Conjugate coding. *ACM SIGACT News*, 15(1):78–88, 1983.
- [5] A. Wyner. The wire-tap channel. *Bell Systems Technical Journal*, 54(8):1355–1387, 1975.



## Table of Contents

Abstract .....	iii
Key Words and Phrases .....	v
Acknowledgements .....	vii
Author’s Assessment of the Project .....	xi
Preface .....	xiii
Outline .....	xvi
Additional Comments .....	xvi
Nomenclature and Conventions .....	xvii
Preface Bibliography .....	xvii
List of Figures .....	xxi
<b>Part 1. Preliminaries .....</b>	<b>1</b>
Chapter 1. Introduction and Literature Survey .....	3
1.1. Aims and Objectives .....	4
1.2. Fundamental Concepts .....	4
1.3. Modelling Quantum Protocols .....	8
1.4. Summary and Conclusion .....	10
Chapter 1 Bibliography .....	10
Chapter 2. The Quantum Cryptographic Protocols BB84, B92, and EPR .....	13
2.1. The BB84 Protocol .....	13
2.2. The B92 Protocol .....	17
2.3. The EPR Protocol .....	21
2.4. Summary and Conclusion .....	23
Chapter 2 Bibliography .....	23
<b>Part 2. Modelling and Analysis of the BB84 Protocol .....</b>	<b>25</b>
Chapter 3. A Model of BB84 in CCS .....	27
3.1. Brief Review of CCS .....	27
3.2. The BB84 Protocol Model .....	28
3.3. Results .....	31
A Note Regarding the Use of CWB–NC .....	31

3.4. Summary and Conclusion .....	31
Chapter 3 Bibliography .....	32
Chapter 4. Analysis of BB84 using PRISM .....	33
4.1. A PRISM Model of BB84 .....	33
4.2. Desired Properties of the Protocol .....	38
4.3. Verification of the PRISM Model .....	40
4.4. Relation of Results to Theory .....	42
4.5. Summary and Conclusion .....	45
Chapter 4 Bibliography .....	45
<b>Part 3. Epilogue .....</b>	<b>47</b>
Chapter 5. QKD-FORM: A Formalism for Describing Quantum Protocols ...	49
5.1. Formal Definition of the QKD-FORM Syntax .....	49
5.2. The Semantics of QKD-FORM, Informally .....	50
Chapter 6. Conclusions and Future Work .....	53
Cumulative Project Bibliography .....	53
Appendix A. PRISM Model of Single-Bit Version of BB84 .....	55
Appendix B. Sample PRISM Output .....	59
Appendix C. Shell Scripts for Processing PRISM Output .....	63
constr.sh .....	63
modelcheck1to64.sh .....	63
retrieveprobs.sh .....	63
retrievetimes.sh .....	64

## List of Figures

1	Polybius, <i>Historiae</i> , Book x, Chapter 43 . . . . .	xiv
2	Polybius, <i>Historiae</i> , Book x, Chapter 44 . . . . .	xiv
1	PROTOCOL <b>BB84</b> (QUANTUM KEY DISTRIBUTION). . . . .	15
2	FORMAL DESCRIPTION OF <b>BB84</b> . An attempt at describing the <b>BB84</b> protocol in QKD-FORM, a formal, programming language – like notation. . . . .	17
3	PROTOCOL <b>B92</b> (QUANTUM KEY DISTRIBUTION). . . . .	18
4	FORMAL DESCRIPTION OF <b>B92</b> . A description of the <b>B92</b> protocol in QKD-FORM, a formal, programming language – like notation. . . . .	20
5	PROTOCOL <b>EPR</b> (QUANTUM KEY DISTRIBUTION). . . . .	22
6	FORMAL DESCRIPTION OF <b>EPR</b> . A description of the <b>EPR</b> protocol in QKD-FORM, a formal, programming language – like notation. . . . .	23
1	The agent <b>A</b> representing Alice. . . . .	29
2	The agent <b>B</b> representing Bob. . . . .	29
3	The interaction of the two channel agents. . . . .	29
1	Time taken to compute probabilities for multiple-bit version with $N = 5$ . . . . .	40
2	Time taken to compute $P_{det}^-$ for key lengths of 5 to 64 bits. . . . .	41
3	Probabilities computed by PRISM for multiple-bit version of <b>BB84</b> with $N = 5$ . . . . .	41
4	Probabilities computed by PRISM for single-bit version of <b>BB84</b> with $N = 64$ . . . . .	42
5	$P_{det}$ versus $N$ , where $5 \leq N \leq 64$ bits. . . . .	42
6	Probability of Not Detecting Eavesdropping (a) and Probability of Not Knowing Whether Eavesdropping has Occurred (b) versus number of bits transmitted. . . . .	43



PART 1

## **Preliminaries**



## CHAPTER 1

# Introduction and Literature Survey

*If anyone thinks he knows Quantum mechanics,  
he doesn't know Quantum mechanics.*

— RICHARD FEYNMAN

Anyone who is not shocked by quantum theory has  
not understood a single word.

— NIELS BOHR

**Q**UANTUM THEORY has puzzled mankind since it was first conceived. Yet it is the most complete and accurate scientific model of the world around us. Research in recent years has made Quantum Theory increasingly relevant to Computer Science. The product of this research is the new field of Quantum Computation and Quantum Information Theory.

Widespread interest in this new field was ignited by the discovery of an efficient *quantum algorithm* for factoring composite integers by Peter Shor [13, 17, 5] in 1994. This is very significant, as the problem of factoring integers cannot be solved efficiently on a classical computer [7]. Most of the public-key cryptographic systems in use today rely heavily on the difficulty of solving this problem. Shor demonstrated that if a quantum computer was ever built, the security of such systems could easily be compromised.

A year later, Lov Grover discovered a quantum algorithm for efficiently searching an unstructured space [13]. The notion of information-processing quantum mechanical devices, called *quantum computers*, thus gained in currency. Physicists had been toying with the idea for a while, but the difficulty of implementing such devices was (and still remains) a major obstacle to progress in this exciting field.

It was several years earlier, however, in 1984, that Charles Bennett and Gilles Brassard had developed quantum cryptography [2, 4]. Quantum cryptography was a major advance in the study of secure systems. For the first time in history, a perfect cryptosystem that was actually feasible had been created.

Quantum computers will be capable of breaking existing cryptographic systems, but they will give us the opportunity to develop new and more powerful ones. Quantum cryptographic systems exploit the randomness inherent in quantum mechanics in order to achieve provably secure communication [17]. When this scheme is implemented successfully, governments, banks and industry (and any individual with the necessary equipment) will all be able to exchange information in perfect secrecy. Recently, it has been proved that quantum cryptography is completely unbreakable, no matter how powerful the attack is.

This thesis presents systematic analyses of quantum cryptographic protocols. The work discussed herein formed the core of an undergraduate computing project. This project was undertaken by the author in the final year of his degree in Computer Systems Engineering at the University of Warwick.

## 1.1. Aims and Objectives

Formal methods for specifying and verifying the behaviour of complex systems have proved very successful in recent years. Such methods are particularly well-suited to the design of computer protocols, and specialist software tools have been developed to aid in this endeavour [6, 14].

Specification formalisms used in the study of concurrency, such as CCS [11] and CSP [14], are among the various methods that can be used for this purpose. Once the behaviour of a protocol has been completely described using the chosen formalism, it can be checked whether it satisfies certain properties. The most important property is that of *correctness*, and when this property is satisfied, the protocol is mathematically guaranteed to operate according to specification.

Verification of these properties is particularly valuable for security protocols, which generally provide, according to [14], the following services to their users:

- secrecy,
- authentication,
- integrity,
- non-repudiation,
- fairness,
- anonymity,
- and availability.

Formalisms such as those discussed previously have not yet been systematically applied to the design and validation of quantum cryptographic protocols. The aim of this work is to make that crucial step. This will have merit for future implementations of quantum cryptography, and will provide a framework within which novel protocols for quantum computer systems can be tested.

Original investigations by R. Nagarajan and S. Gay [12] have involved the use of CCS for specifying quantum protocols. The software tool CWB-NC was used to check whether the BB84 protocol worked according to CCS specifications. These investigations have been successful, and the ultimate goal of this project is to further that work by experimenting with other modelling languages and software tools, especially the PRISM model-checker developed at the University of Birmingham [8].

The remainder of this chapter presents some fundamental concepts necessary for an understanding of quantum cryptographic protocols and their modelling.

## 1.2. Fundamental Concepts

This section attempts to review the concept of cryptography, to define the notion of a cryptographic protocol, and then to describe *quantum* protocols. The problem of key distribution is solved with the use of quantum protocols, and this is explained in detail. All the associated terminology is introduced.

### 1.2.1. Cryptography

Cryptography is the term used to describe the process of exchanging secret messages. The very term *cryptography* stems from the Greek words *κρυπτός*, meaning *hidden*, and *γραφή*, meaning *writing*. The importance of cryptography has been recognised since ancient times, and Julius Caesar even had a cryptosystem of his own.

The term *encryption* is used to describe the process by which a simple message (the *plaintext*) is made unintelligible. The result of encryption is called the *ciphertext*, and a special *key* needs to be used in order to recover the plaintext from the ciphertext. *Cryptosystem* is the collective term used to refer to a particular method of encryption and decryption. Extensive treatments of cryptography and its mathematical foundations are given in [16, 4].

### 1.2.2. Cryptographic Protocols

In modern computer networks, cryptography is implemented using special *protocols*. Protocols in general are defined as procedures governing the exchange of data between two agents (e.g. people or computers). The precise definition of a protocol given in [6] is:

- ▶ **Definition 1.1 (Protocols).** *Protocols are sets of rules that govern the interaction of concurrent processes in distributed systems.*

A *cryptographic protocol* or *security protocol* is a particular kind of protocol designed to ensure that the exchange of data is secure. The following is from [14]:

- ▶ **Definition 1.2 (Security Protocols).** *The goals of security protocols are to provide various security services across a distributed system.*

The security services mentioned in the above definition are precisely those listed on page 4.

### 1.2.3. Quantum Cryptography

Quantum cryptography is a novel kind of cryptography that involves the use of quantum phenomena in its implementation. It is not in itself a new cryptosystem, but rather a *principle* for designing new cryptosystems. Quantum cryptography uses subatomic particles to encode data. The properties of such particles allow any interference with the data to be detected. An enemy cryptanalyst or *eavesdropper* who tries covertly to obtain the data being exchanged between two parties is always detected using this method. Quantum cryptography was originally presented in the *Scientific American* (reprinted with corrections in [2]).

### 1.2.4. Quantum Protocols

Security protocols that implement the technique of quantum cryptography include BB84, B92, and the EPR protocol. These protocols will be discussed in detail in Chapter 2. A *quantum protocol* is defined as follows:

- ▶ **Definition 1.3 (Quantum Protocol).** *A quantum (cryptographic) protocol is a data communications procedure which employs quantum phenomena and is designed to ensure secure communication.*

Quantum protocols such as BB84 were originally developed for the exchange of cryptographic keys only. That is, the first quantum protocols were not designed for communicating encrypted data, but rather to handle the keys needed for two parties to encrypt their data. The protocol proposed recently by Kumar, Yuen *et al.* [1] is the first to tackle exchange of encrypted *data* with quantum effects.

The quantum protocol BB84 has been shown to be unconditionally secure, i.e. immune against all possible attacks [3]. If such a protocol is used to exchange cryptographic keys, the keys are guaranteed to be secure. A classical cryptosystem

that makes use of these keys can then be used to communicate data in secrecy. Indeed, a classical cryptosystem that is *perfect*, such as the *one-time pad*, can be used once the keys have been exchanged. This means that provably unbreakable cryptography is possible.

### 1.2.5. The Qubit

In classical computing, the unit of information is the *bit*. All computer data is represented by sets of bits. Designers of communications networks apply the principles of information theory and coding to make transmission of bits as efficient as possible. Classical cryptography involves manipulation of bits too. The quantum mechanical counterpart of the bit is the *qubit*. Quantum cryptography involves representing secret data and keys by sequences of qubits.

A classical bit corresponds to a single binary state, represented by a value of 1 or a value of 0. A bit is always well-defined, since it can have exactly one of these two values. This is in stark contrast to the value of a qubit: a qubit may have a value of  $|1\rangle$ , a value of  $|0\rangle$ , or anything “in between”. Notice the special notation for quantum bits: the  $|\ \rangle$  symbol is called a *ket*. A qubit does not take on only one of two values, as a bit does, but one of infinitely many values.

A qubit is formally defined as follows [5]:

- **Definition 1.4 (Qubit).** *A qubit is a vector  $|\psi\rangle$  such that*

$$|\psi\rangle = a \cdot |0\rangle + \beta \cdot |1\rangle$$

*The parameters  $a$  and  $\beta$  are complex numbers that satisfy  $|a|^2 + |\beta|^2 = 1$ .*

Therefore, the value of a qubit is generally some combination of the two basic states  $|0\rangle$  and  $|1\rangle$  determined by the parameters  $a$  and  $\beta$ . The combined state  $|\psi\rangle$  is called the *superposition* of  $|0\rangle$  and  $|1\rangle$ .

A qubit manifests itself in the physical world in the form of any subatomic particle, e.g. an electron, or a photon. The most important feature of such a particle is that, while unobserved, it will remain in some state  $|\psi\rangle$  with arbitrary  $a$  and  $\beta$ ; *when its value is measured (the particle is observed), it will collapse into one of the two basic  $|0\rangle$  or  $|1\rangle$  states.* That is, attempting to measure the state of a qubit irrevocably alters that state. This property is exploited in quantum cryptography in order to detect the presence of eavesdropping; if an unobserved quantum state is sent over a communications channel, an eavesdropper will change that state simply by observing it.

### 1.2.6. Photon Polarisation and Polarisation Bases

So quantum cryptography involves manipulation of qubits — and qubits can be in any one of infinite states. But what does this mean physically? In section 1.2.3 it was stated that quantum cryptography makes use of subatomic particles. Photons and electrons are good examples of candidates for physical qubits. Some property of these particles has to be used to represent *state*. An appropriate property for encoding the state of a qubit the *polarisation of a photon*.

A *photon* is known to most people as a particle of light. At certain frequencies, however, a photon will exhibit *wave-like* behaviour. This dual nature of photons (and other subatomic particles) is known to physicists as *wave-particle duality*. In actual fact, a photon is best thought of as an electromagnetic wave — a wave produced by an electric field perpendicular to a magnetic field.

In a three-dimensional coordinate system, this corresponds to an electric field vector lying in a given plane, and a magnetic field vector in a plane exactly  $90^\circ$  away from the former. The angle of the plane in which the electric field lies is called the photon's *polarisation*.

There are devices that can be used to fix the polarisation of a given photon; This is exactly what is needed to encode a quantum state. In direct analogy to the way in which a transistor can be set in a desired state (corresponding to a classical 0 or 1) using appropriate input currents, a photon's polarisation can be set to a particular angle (corresponding to a quantum state  $|\psi\rangle$ ).

A *basis* used to encode quantum bits to photons is the pair of polarisation angles that correspond to the values of those bits. A slightly more general definition is the following:

- **Definition 1.5 (Basis).** *The basis or quantum alphabet maps a pair of binary values to the states of a quantum-mechanical system, such as the polarization angles of a photon.*

$$\beta : (0, 1) \mapsto (\vartheta, \varphi)$$

If, in a given quantum cryptographic system, the *rectilinear basis* is used, then 0 is encoded as a photon with a polarisation of  $0^\circ$ , and 1 is encoded as a photon with a polarisation of  $90^\circ$ . When a *diagonal basis* is used instead, the corresponding polarisation angles that are used are  $45^\circ$  and  $135^\circ$ .

- **Definition 1.6 (Rectilinear Basis).** *A photon is encoded with a rectilinear basis, denoted by  $\boxplus$ , if a 0 is mapped to a polarisation of  $0^\circ$  and a 1 is mapped to a polarisation of  $90^\circ$ . Formally, the rectilinear basis is the mapping:*

$$\boxplus : (0, 1) \mapsto (0^\circ, 90^\circ)$$

- **Definition 1.7 (Diagonal Basis).** *A photon is encoded with a diagonal basis, denoted by  $\boxtimes$ , if a  $|0\rangle$  is mapped to a polarisation of  $45^\circ$  and a  $|1\rangle$  is mapped to a polarisation of  $135^\circ$ . Formally, the diagonal basis is the mapping:*

$$\boxtimes : (0, 1) \mapsto (45^\circ, 135^\circ)$$

An important property of bases is *orthogonality*. A particular basis is orthogonal if the two polarization angles that it involves are perpendicular to each other. This has great physical significance since it affects the method in which photons are *measured*.

- **Definition 1.8 (Orthogonal Basis).** *A basis  $\beta$  defined as:*

$$\beta : (0, 1) \mapsto (\vartheta, \varphi)$$

*is said to be orthogonal if and only if:*

$$\vartheta \perp \varphi$$

An example of a *non-orthogonal* basis is

$$\beta : (0, 1) \mapsto (-20^\circ, 20^\circ)$$

(clearly  $\vartheta \not\perp \varphi$ ). It is obvious that:

- **Claim 1.1.** *The rectilinear basis ( $\beta = \boxplus$ ) and the diagonal basis ( $\beta = \boxtimes$ ) are orthogonal bases.*

Given a photon which encodes a particular bit, how do we decode it? Decoding in this context corresponds to performing a physical *measurement*. Measuring the state of a photon is done with respect to a particular polarisation basis. If the same basis that was used to encode a bit to a photon is used to decode it, then the result is guaranteed to be correct. However, if a different basis is used for decoding, the result of the measurement is *random*. There is only 50% chance of obtaining the correct bit value in this case. This statement is expressed formally below:

- ▶ **Claim 1.2 (Photon Measurement).** *Let a bit value  $\mathbf{d}$  be encoded to a photon using a polarisation basis  $\beta$ . The resulting photon  $\mathbf{P}$  is subsequently measured using a polarisation basis  $\beta'$ . If  $\beta = \beta'$ , then the measurement of  $\mathbf{P}$  is guaranteed to produce the original bit  $\mathbf{d}$ . If  $\beta \neq \beta'$ , then the measurement of  $\mathbf{P}$  will produce either the original bit  $\mathbf{d}$  or its complement  $\overline{\mathbf{d}}$  with equal probability.*

Another important case of photon measurement is when a non-orthogonal basis is used:

- ▶ **Claim 1.3 (Measurement Using Non-Orthogonal Basis).** *Let a bit value  $\mathbf{d}$  be encoded to a photon using a non-orthogonal polarisation basis  $\beta$ . Quantum Theory predicts that there is no one experiment that will unambiguously distinguish between the two polarization states of such a photon (see [10]).*

\*\*\*

Now that the basic concepts associated with quantum cryptography have been discussed, the core focus of this project will be presented: modelling the behaviour of quantum protocols and checking their validity and level of security.

### 1.3. Modelling Quantum Protocols

Designing cryptographic protocols for classical computers is complicated enough; Several methods have been proposed, and the most recent work in this area is [14].

Quantum protocols are far more difficult to design. The reader can appreciate the additional level of complexity that quantum protocols admit from the previous section. Randomness is inherent in quantum systems and quantum protocols are no different. Whereas a classical protocol is deterministic and all the interactions involved are predictable, a quantum protocol can only be modelled if the *probability* of a particular interaction taking place is considered. For example, observing a quantum state (such as that carried by a rectilinearly polarized photon) will not yield the correct value for sure. It will yield a  $|0\rangle$  with some probability  $\mathcal{P}$  and a  $|1\rangle$  with probability  $1 - \mathcal{P}$ .

The first attempt to formally model quantum protocols [12] did not include probabilities. Nevertheless, the use of the specification language CCS was sufficient in order to model the simpler aspects of these. There is a way of expressing events happening at random in CCS — but this does not include the probabilities with which these events occur. It is clear that, in order to completely describe and design quantum protocols, a more powerful specification method is needed.

What is more, once quantum protocols can be fully described in a specification language, a tool is required that can verify whether a given implementation

conforms to the specification. It seems that the most appropriate language — for both specifying and verifying quantum protocols — is PRISM.

### 1.3.1. The PRISM System

PRISM is a specialized piece of software developed at the University of Birmingham [8] especially to aid in the specification and verification of complex systems<sup>1</sup>. It automatically verifies properties of *probabilistic models*. Probabilistic models are labelled transition systems which incorporate information about the likelihood of transitions between states occurring. There are three main types of probabilistic models:

- (1) Discrete–Time Markov Chains (DTMCs),
- (2) Markov Decision Processes (MDPs),
- (3) and Continuous–Time Markov Chains (CTMCs).

These variations are described in more detail below<sup>2</sup>.

*Discrete–Time Markov Chains* define the probability of making a transition from one state to another. DTMCs can be used to model either a single probabilistic system or several such systems which operate in synchrony. A DTMC is defined formally as follows:

- **Definition 1.9 (Discrete–Time Markov Chain).** *A DTMC is a tuple  $(S, \bar{s}, \mathbf{P}, L)$  where  $S$  is a finite set of states,  $\bar{s}$  is the initial state,  $\mathbf{P} : S \times S \mapsto [0, 1]$  is the transition probability matrix and  $L : S \mapsto 2^{AP}$  is the labelling function.*

Note that, in the definition above, an element  $\mathbf{P}_{ss'}$  in the transition probability matrix denotes the probability of making a transition from state  $s$  to state  $s'$ . The labelling function is used to attach properties of interest (expressed as atomic propositions from a set  $AP$ ) to particular states.

A *Markov Decision Process* is a generalisation of a DTMC, and is capable of describing both probabilistic and nondeterministic behaviour. Nondeterministic systems differ from probabilistic systems in that, the exact probabilities of certain transitions are not known, or are known but irrelevant. Also, nondeterministic models can be thought of as sets of probabilistic systems operating in parallel. The formal definition of a MDP is:

- **Definition 1.10 (Markov Decision Process).** *A MDP is a tuple  $(S, \bar{s}, \text{Steps}, L)$  where  $S$  is a finite set of states,  $\bar{s} \in S$  is the initial state,  $\text{Steps} : S \mapsto 2^{\text{Dist}(S)}$  is the transition function and  $L : S \mapsto 2^{AP}$  is the labelling function.*

The definition of a MDP differs from that of a DTMC in that, instead of a transition matrix, a *transition function* is used. For a given state  $s \in S$ , the elements of  $\text{Steps}(S)$  denote nondeterministic choices available in that state; each such choice is a probability distribution (belonging to the set of probability distributions  $\text{Dist}(S)$ ) which gives the probability of making a transition to any other state.

Finally, *Continuous–Time Markov Chains* differ from DTMCs in that transitions occur in real time, not in discrete time steps. A CTMC is defined in a similar manner to DTMCs and MDPs but involves a *transition rate matrix*, which states the rates

<sup>1</sup>PRISM can be downloaded from the following URL: <http://www.cs.bham.ac.uk/~dxp/prism/>. The version used for this work was PRISM 1.2 for Linux.

<sup>2</sup>The author wishes to acknowledge the help of Dr. David Parker, who kindly provided material from his doctoral dissertation, upon which this section is based.

at which transitions occur as opposed to their probabilities. CTMCs are not considered further in this thesis.

The aforementioned types of models are suitable for describing the behaviour of cryptographic protocols that are probabilistic, such as quantum protocols. This matter is discussed in section 4.1. In order to verify whether such models satisfy certain properties, the properties have to be expressed in a formal manner. The notation used in the PRISM tool to verify DTMCs and MDPs is named PCTL, which is an abbreviation for Probabilistic Computational Tree Logic.

In order to construct and verify a model in PRISM, two input files are required:

- (1) A *system description file*, consisting of a description of the model in the PRISM language. The type of the model is declared at the beginning of the file, and may be probabilistic (i.e. a DTMC), nondeterministic (i.e. a MDP), or stochastic (i.e. a CTMC). The rest of the file consists of variable declarations, and a listing of each of the agents in the model (termed *modules* in PRISM). Each module is made up of a set of states and transitions between them.
- (2) A *properties file*, consisting of a list of properties which the model must satisfy. The properties are expressed in PCTL for probabilistic and nondeterministic models.

### 1.3.2. The Concurrency Workbench of the New Century (CWB-NC)

The Concurrency Workbench of the New Century (CWB-NC<sup>3</sup>) is the tool used in [12] to verify the quantum protocol BB84. It is an interpreter for the CCS formalism, originally designed by Robin Milner [11] for the description of concurrent systems.

The use of the CWB-NC for specifying quantum protocols is discussed in Chapter 3. It must be noted that this tool is somewhat simpler than PRISM, but it does not allow probabilities to be included in system models.

## 1.4. Summary and Conclusion

This chapter has supplied the reader with all the background relevant to the specification and verification of quantum cryptographic protocols. Firstly, the origins of this project were presented. Then an account was given of previous work and the aims and objectives of the project. The technicalities of cryptography and quantum information, and the nomenclature involved, were detailed. Finally, the modelling and analysis techniques to be applied to quantum protocols were described.

It is well-understood that the techniques used in this project are timely and novel. Formal specification and verification is generally accepted as a useful and successful practice in industry. There already exist industrial implementations of quantum cryptography and it is likely that there will be several more in years to come. The methods described herein will provide a sound theoretical basis for designing and testing such systems.

## Chapter 1 Bibliography

- [1] G. A. Barbosa, E. Corndorf, P. Kumar, H. P. Yuen, G. D' Ariano, M. Paris, and P. Perinotti. Secure communication using coherent states. *quant-ph/0210089*, 2002.

---

<sup>3</sup>CWB-NC can be downloaded from <http://www.cs.sunysb.edu/~cwb/>

- [2] C. H. Bennett, G. Brassard, and A. K. Ekert. Quantum cryptography. *Scientific American Special Issue: The Computer in the 21st Century*, 1995.
- [3] E. Biham, M. Boyer, P. O. Boykin, T. Mor, and V. Roychowdhury. A proof of the security of quantum key distribution. [quant-ph/9912053](#), December 1999.
- [4] G. Brassard. *Modern Cryptology: A Tutorial*. LNCS 325. Springer-Verlag, 1988.
- [5] J. Gruska. *Quantum Computing*. McGraw-Hill International, 1999.
- [6] G. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.
- [7] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, third edition, 1998.
- [8] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic Symbolic Model Checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Computer Performance Evaluation (TOOLS'02)*, pages 200–204. Springer-Verlag, 2002.
- [9] S. Lipschutz. *Probability*. Schaum's Outlines. McGraw-Hill, 1965.
- [10] S. J. Lomonaco, Jr. A quick glance at quantum cryptography. [quant-ph/9811056](#), 1998.
- [11] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
- [12] R. Nagarajan and S. Gay. Formal verification of quantum protocols. [quant-ph/0203086](#), March 2002.
- [13] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, first edition, 2002.
- [14] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.
- [15] L. Takács. *Stochastic Processes: Problems and Solutions*. Methuen & Co. Ltd., 1960.
- [16] D. Welsh. *Codes and Cryptography*. Oxford Science Publications. Oxford University Press, 1998.
- [17] C. P. Williams and S. H. Clearwater. *Ultimate Zero and One*. Copernicus (Springer-Verlag), 2000.



## CHAPTER 2

# The Quantum Cryptographic Protocols BB84, B92, and EPR

*[Quantum teleportation] is quite similar to transubstantiation, the process whereby a person will suddenly dematerialize and rematerialize somewhere else in the world. This is not a bad way to travel, although there is usually a half-hour wait for luggage.*

— WOODY ALLEN

*The human mind treats a new idea the way the body treats a strange protein – it rejects it.*

— PETER MEDAWAR

**T**HE AIM of this chapter is to present the three main quantum cryptographic protocols that have been proposed to date. All of these protocols implement quantum key distribution (QKD). The sender and receiver, just as in classical cryptography, are conventionally called *Alice* and *Bob*. Using these protocols, Alice and Bob are able to establish a common cryptographic key. The first protocol (BB84) will be systematically analysed and modelled in later chapters.

### 2.1. The BB84 Protocol

The BB84 protocol is named after its two inventors, Charles Bennett and Gilles Brassard, who published it in 1984. The key emerges as the protocol proceeds, starting from an initial random bit sequence.

There are two channels of a different nature separating Alice and Bob. One channel is private and is made to carry quantum bits; for this reason, it is termed the *quantum channel*. The other channel is classical and is publicly accessible: it could equally well be a telephone line or a radio broadcast. In fact, *any* data exchanged over the second channel is insecure and open to everyone. Curiously, this feature does not affect the overall security of BB84. None of the information exchanged over this channel is of use to an eavesdropper.

BB84 takes place in two phases, the first over the quantum channel, the second over the public, classical channel. In the first phase, Alice and Bob exchange a random set of qubits (typically photons over an optical fibre). In the second phase, they discuss some of the measurements that they made.

An eavesdropper, *Eve*, is assumed to have access to both the quantum channel and the classical channel. Furthermore, Eve is assumed to have equipment just as powerful as Alice's and Bob's available. Due to the properties of quantum bits, Eve's presence will not go completely unnoticed. Alice and Bob will take measures to minimize the amount of valid information that Eve can obtain about their secret key.

In the first phase of the protocol, Alice chooses a basis with which to encode a bit of data at random. When Bob receives the photons sent by Alice, he has to

decode each using *another* basis chosen at random. In many cases, Bob's choice of basis will be different from Alice's, and then he is bound to get an incorrect bit of data (due to Theorem 1.2). It is found by calculation that there is a 25% chance that each measurement Bob makes is wrong.

The second phase of the protocol allows Alice and Bob to compare their measurements and to detect whether eavesdropping has occurred. In those cases where Bob made an incorrect measurement, he and Alice discard the corresponding bits. The final step BB84.2.5 is the most important part of the protocol. In this step, a selection of bits for which both Alice and Bob have used the same basis are sent by Alice over the public channel. Normally, the corresponding bits in Bob's sequence should agree. In other words, since Alice and Bob used the same basis for encoding and decoding the bits in  $TFK_A$  and  $RCK_B$ , it should be that  $TFK_A = TFK_B$ . If not, it means that an eavesdropper has measured the photons on the channel, and sent random substitutes to Bob. An eavesdropper cannot measure the photons and simultaneously send an identical copy to Bob. This is the property of *non-cloneability* of quantum states and is what makes eavesdropping detectable in BB84.

The BB84 protocol is shown in detail in Figure 2.1. The version of the protocol shown in the figure is capable of dealing with errors. Physicists have been able to devise clever error-correction techniques; it is, in fact, possible to distinguish channel errors from errors induced by eavesdropping. Modelling quantum error correction procedures in detail is an area for future work. For reasons of simplicity, quantum error correction only discussed briefly in section 2.1.

Figure 2.1 deserves some explanation. All the protocols described in this chapter will be displayed in the same format. Each step of the protocol has a unique number, e.g. BB84.2.1 which will be used for reference purposes. Furthermore, each step has a short description, somewhat in the style of Knuth [2].

\*\*\*

The above presentation of BB84 raises some interesting questions:

- What about error correction?
- What is privacy amplification (see (BB84.2.4))?
- How can Eve try to attack the protocol?

A short discussion of these issues is given below.

**Error Correction.** In quantum key distribution (QKD), discrepancies between sent and received bits are assumed to be a consequence of eavesdropping. Also, the protocols will work only if the quantum channel error rate is above a certain threshold. Therefore, care needs to be taken to ensure that eavesdropping is distinguished from channel errors. For this purpose, a quantum key distribution protocol normally includes an *error correction* procedure and a *privacy amplification* procedure. Since QKD is particularly susceptible to errors, the use of quantum error correction is necessary.

**Privacy Amplification.** As Alice and Bob perform steps to reduce errors, they must not disclose any information about the key itself to a potential eavesdropper. Privacy amplification is the process by which Alice and Bob minimize the amount of mutual information such a third party could ever gain about the key. In the BB84 protocol, *some* of the original bits will be correctly decoded by an eavesdropper. Consequently, the key that results in the presence of an eavesdropper is only

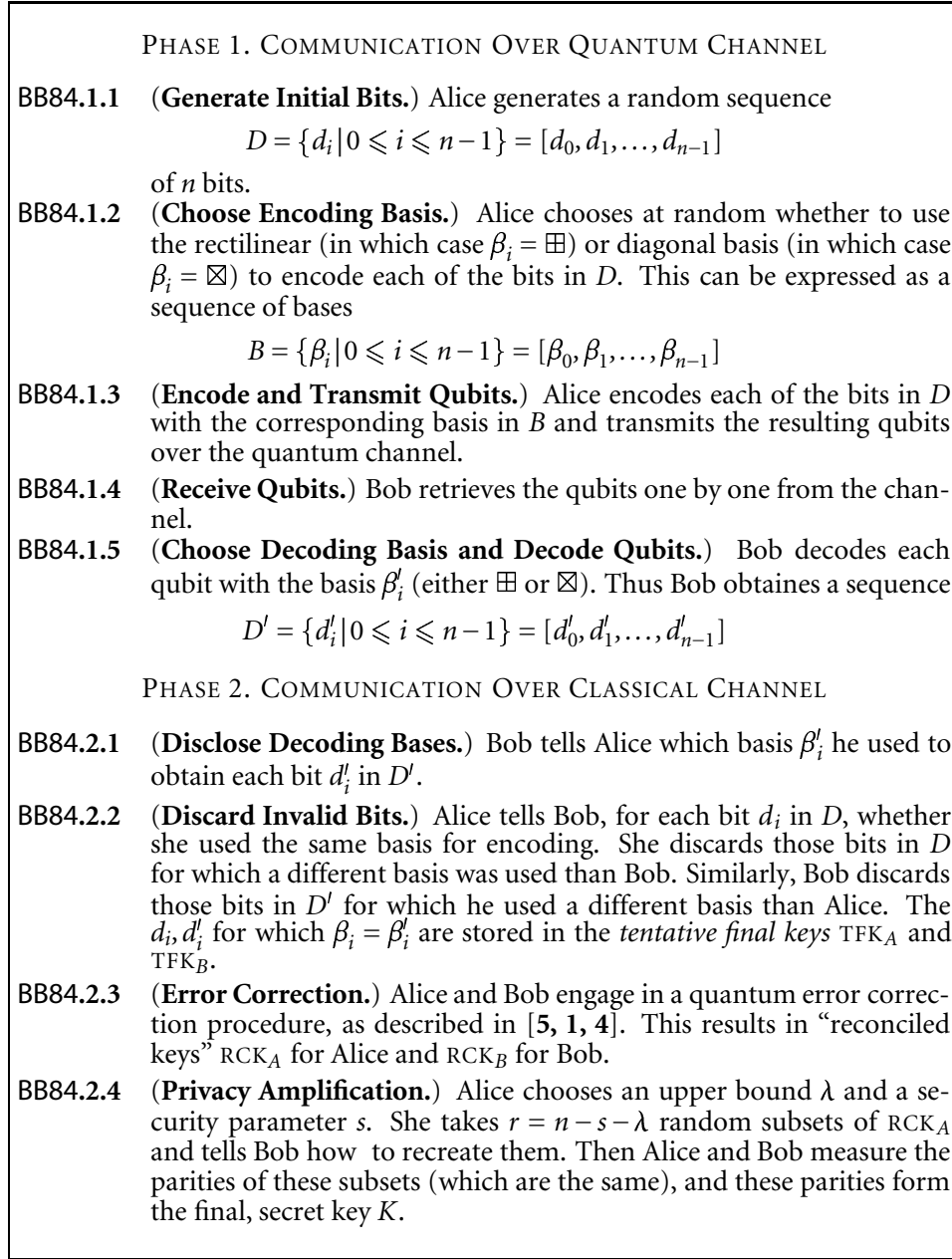


FIGURE 2.1. PROTOCOL BB84 (QUANTUM KEY DISTRIBUTION).

partially secure. The *tentative final keys*

$$\text{TFK}_A \stackrel{?}{=} \text{TFK}_B$$

are subjected to error correction, thus resulting in the *reconciled keys*

$$\text{RCK}_A \stackrel{?}{=} \text{RCK}_B.$$

An eavesdropper is likely to know some of the bits in the reconciled key. Privacy amplification ensures that those bits known to the eavesdropper are not used in the final secret key  $K$ , common to Alice and Bob.

Privacy amplification consists of the following steps. Alice and Bob decide on an upper bound  $\lambda$  on the number of bits likely to be known to the eavesdropper. They also decide on a *security parameter*  $s$ . Assuming their key consists of  $n$  bits, they select  $(n - \lambda - s)$  random subsets of the reconciled key, without revealing their contents. The parities of these subsets form the final secret key. According to [1], the probability that an eavesdropper has *any* information about the parity of *any one* of these subsets is:

$$\mathcal{P}_{kp} \leq \frac{2^{-(n-k-1)}}{\ln 2}$$

**Collective and Coherent Attacks.** Various kinds of attacks on BB84 and similar protocols have been suggested to date. In particular, eavesdropping on a quantum channel can be *opaque* or *translucent*. The use of translucent eavesdropping, combined with a so-called *coherent attack*, is the most general way of attacking a QKD protocol. A coherent attack is a special case of a *collective attack*, which is based on the idea of performing measurements on the channel *after* the final key has been decided. The eavesdropper attaches a special *probe* to particles sent over the quantum channel; these probes are stored in a quantum memory until Alice and Bob perform error correction and privacy amplification. A maximal amount of information about the final key can be obtained by correlating this data. When a probe is attached to *all* the bits transmitted over the channel, we have a coherent attack.

### 2.1.1. A Formal Description of BB84

It is necessary to describe BB84 in more formal terms. In particular, expressing the protocol's behaviour in a formal programming language – like notation will facilitate writing models of its behaviour in CCS and PRISM.

Figure 2.2 shows a description of BB84 in a simple notation (“QKD-FORM”) that includes conditional statements (**if... end if**) and loops (**for... end for**). Alice and Bob's actions are written in a CCS-like syntax (i.e. *Agent.action*). For example,  $A.encode(s_i, \beta_i)$  denotes the action *encode* being performed by Alice ( $A$ ) on the data bit  $s_i$  with the basis  $\beta_i$ . The matching descriptions, e.g. (**Transmit Photons**), are shown to the right of the actions.

The actions themselves (*encode, decode, send, qsend, keep, discard, abort*) are self-explanatory. Note the distinction between *qsend* (“send a photon over the quantum channel”) and *send* (“send a bit or message over the classical channel”).

A few more words are in order concerning the notation of Figure 2.2. The actual data being sent between Alice ( $A$ ) and Bob ( $B$ ) consists of photons  $P_i$ , and bits  $s_j$  or messages (“correct”/“invalid”) over the classical channel. Each photon is the result of encoding Alice's data bit  $s_i$  using either the rectilinear basis (i.e.  $\beta_i = \boxplus$ ) or the diagonal basis (i.e.  $\beta_i = \boxtimes$ ). The messages “correct” and “invalid” indicate the success or failure of a comparison, respectively: when a “invalid” is received, it means that the current bit is invalid. The notation is called QKD-FORM and is quite arbitrary, but is a suitable formalism for the description of quantum cryptographic protocols. It will be used in subsequent descriptions of other protocols and will be summarized in Chapter 5.

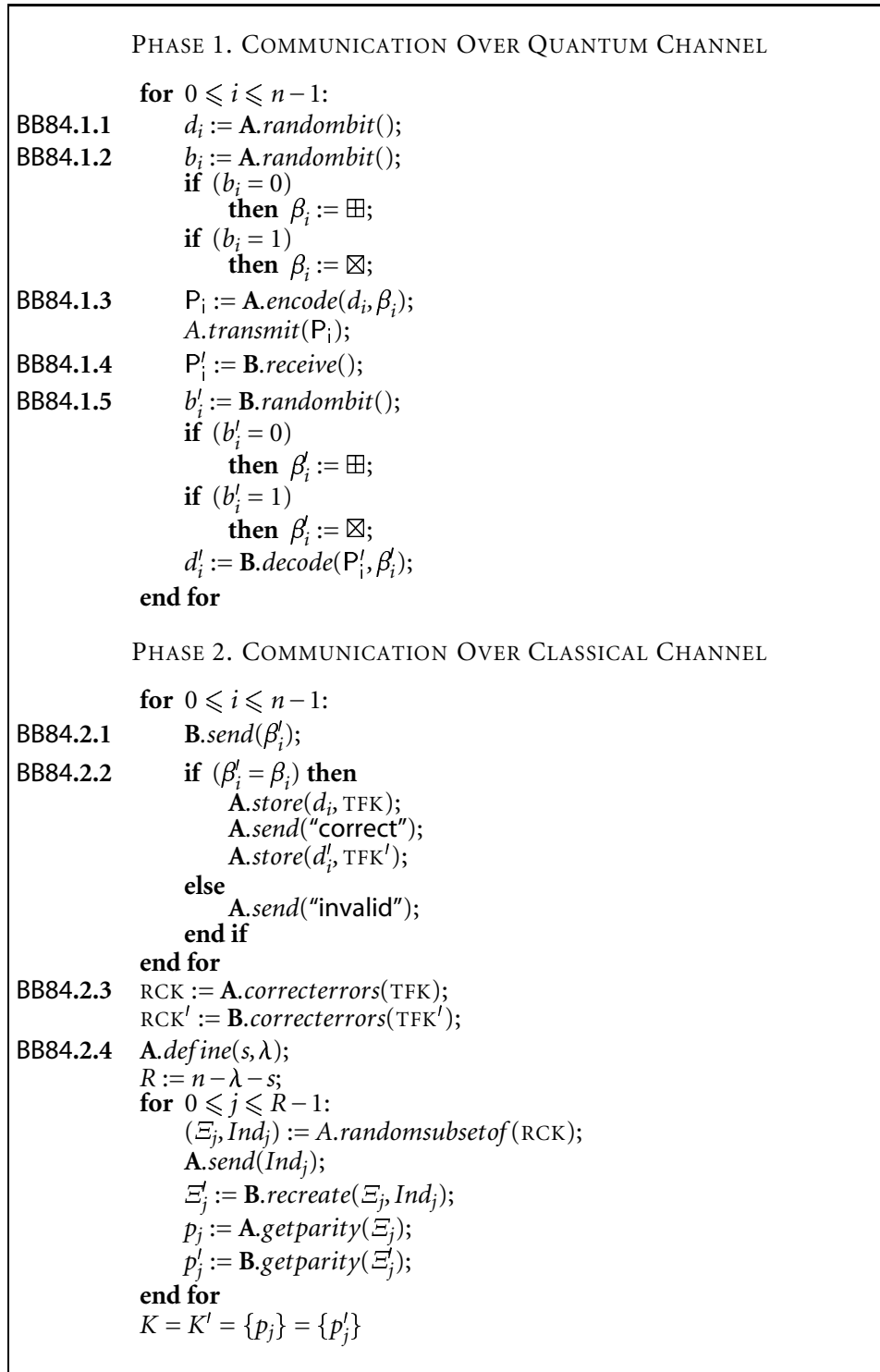


FIGURE 2.2. FORMAL DESCRIPTION OF BB84. *An attempt at describing the BB84 protocol in QKD-FORM, a formal, programming language – like notation.*

## 2.2. The B92 Protocol

Soon after BB84 was published, Charles Bennett realized that it was not necessary to use *two* orthogonal bases for encoding and decoding. It turns out that a

single *non-orthogonal* basis can be used instead, without affecting the security of the protocol against eavesdropping. This idea is used in the B92 protocol, which is otherwise identical to BB84. B92 is detailed in Figure 2.3 and expressed in QKD-FORM in Figure 2.4.

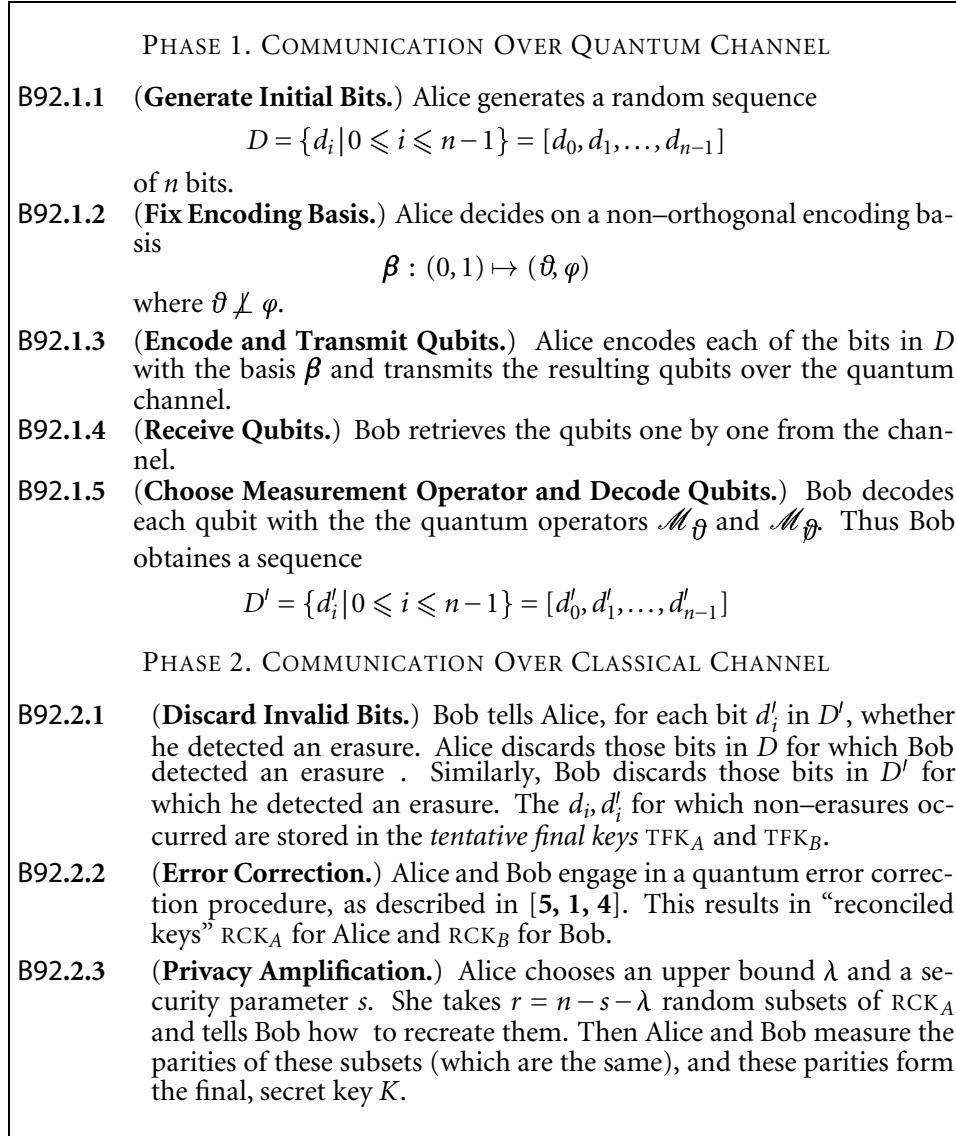


FIGURE 2.3. PROTOCOL B92 (QUANTUM KEY DISTRIBUTION).

Because of Theorem 1.3, neither Bob nor Eve can unambiguously decode all the bits on the quantum channel. However, Bob can perform two separate measurements using the quantum projection operators:

- (1)  $\mathcal{M}_\vartheta = 1 - |\vartheta\rangle\langle\vartheta|$
- (2)  $\mathcal{M}_\varphi = 1 - |\varphi\rangle\langle\varphi|$

This assumes that the non-orthogonal basis is defined as

$$\beta : (0, 1) \mapsto (\vartheta, \varphi)$$

By performing measurements using the operators  $\mathcal{M}_\theta$  and  $\mathcal{M}_{\theta^\perp}$ , Bob either detects Alice's transmitted bit correctly, or a random result known as an *erasure*. The correct measurements are termed *non-erasures*.

The second phase of B92 is the same as that of BB84, and the bits for which Bob received erasures are discarded from both Alice and Bob's sequences. Clearly, Eve's presence is evident from a particularly high error rate in Bob's raw key.

It is interesting to note that, in B92, the probability of Bob receiving Alice's transmission correctly is [3]:

$$\mathcal{P}_{COR} = \frac{1 - \cos^2(2\theta)}{2}, \quad 0 < \theta < \frac{\pi}{4}$$

```

    PHASE 1. COMMUNICATION OVER QUANTUM CHANNEL
    for  $0 \leq i \leq n-1$ :
    B92.1.1  $d_i := \mathbf{A.randombit}()$ ;
    B92.1.2  $\vartheta := \mathbf{A.randomangle}()$ ;
    B92.1.3  $P_i := \mathbf{A.encode}(d_i, \vartheta)$ ;
     $\mathbf{A.transmit}(P_i)$ ;
    B92.1.4  $P'_i := \mathbf{B.receive}()$ ;
    B92.1.5  $b'_i := \mathbf{B.randombit}()$ ;
    if ( $b'_i = 0$ )
    then  $op := \mathcal{M}\mathcal{G}$ ;
    if ( $b'_i = 1$ )
    then  $op := \mathcal{M}\mathcal{H}$ ;
     $d'_i := \mathbf{B.decode}(P'_i, op)$ ;
    end for

    PHASE 2. COMMUNICATION OVER CLASSICAL CHANNEL
    for  $0 \leq i \leq n-1$ :
    B92.2.1  $\mathbf{B.send}(\beta'_i)$ ;
    B92.2.2 if ( $\beta'_i = \beta_i$ ) then
     $\mathbf{A.store}(d_i, \text{TFK})$ ;
     $\mathbf{A.send}(\text{"correct"})$ ;
     $\mathbf{A.store}(d'_i, \text{TFK}'_i)$ ;
    else
     $\mathbf{A.send}(\text{"invalid"})$ ;
    end if
    end for
    B92.2.3  $\text{RCK} := \mathbf{A.correcterrors}(\text{TFK})$ ;
     $\text{RCK}' := \mathbf{B.correcterrors}(\text{TFK}'_i)$ ;
    B92.2.4  $\mathbf{A.define}(s, \lambda)$ ;
     $R := n - \lambda - s$ ;
    for  $0 \leq j \leq R-1$ :
     $(\Xi_j, \text{Ind}_j) := \mathbf{A.randomsubsetof}(\text{RCK})$ ;
     $\mathbf{A.send}(\text{Ind}_j)$ ;
     $\Xi'_j := \mathbf{B.recreate}(\Xi_j, \text{Ind}_j)$ ;
     $p_j := \mathbf{A.getparity}(\Xi_j)$ ;
     $p'_j := \mathbf{B.getparity}(\Xi'_j)$ ;
    end for
     $K = K' = \{p_j\} = \{p'_j\}$ 

```

FIGURE 2.4. FORMAL DESCRIPTION OF B92. A description of the B92 protocol in QKD-FORM, a formal, programming language-like notation.

### 2.3. The EPR Protocol

An entirely different class of quantum cryptographic protocols was suggested by Artur Ekert in 1991: *entanglement-based QKD protocols*. These protocols use so-called EPR pairs of particles in order to encode the bits of a key. EPR stands for *Einstein, Podolsky and Rosen*, who presented a famous paradox in their 1935 paper. The EPR paradox concerns special pairs of particles which are *entangled*, or correlated, with each other. These particles may typically be at a distance from each other, but when one of them is measured, the result of measuring the other is known. This strange behaviour was referred to as “spooky action at a distance”.

In order to account for this, the three physicists suggested the existence of *hidden variables*, which quantum theory had not accounted for. On this premise, they claimed that quantum theory was incomplete. However, in 1964, J. Bell discovered a way of proving whether a given system and/or theory involved hidden variables: hidden-variable theories have to satisfy the so-called *Bell inequality*. It was found that quantum theory does not.

Nevertheless, EPR pairs are still important and can be created using laboratory equipment. Ekert proposed that entangled particles could be used to detect eavesdropping in a quantum cryptographic system; if an eavesdropper measured one of the entangled particles, this would inevitably affect the other one. In the EPR protocol, the eavesdropper is treated as a hidden variable in the exchange of EPR pairs between Alice and Bob. Bell’s inequality can then be used to detect Eve’s presence.

\*\*\*

Consider an entangled pair of photons (P1, P2). If the polarization of P1 is measured, then the polarization of P2 is known and vice versa. To put this in more formal terms, if the mixed quantum state of the EPR pair is given by (here subscript <sub>1</sub> refers to P1, subscript <sub>2</sub> refers to P2):

$$|\Omega\rangle = \frac{1}{\sqrt{2}} \cdot (|\vartheta\rangle_1 |\vartheta'\rangle_2 - |\vartheta'\rangle_1 |\vartheta\rangle_2)$$

then measuring P1 will either give:

- a polarization angle  $\vartheta$  for P1 and a polarization  $\vartheta'$  for P2, or
- a polarization angle  $\vartheta'$  for P1 and a polarization  $\vartheta$  for P2.

In the EPR protocol, three encoding and decoding bases are used:

$$\begin{aligned} \beta_0 & : (0, 1) \mapsto (\vartheta, \vartheta') \\ \beta_1 & : (0, 1) \mapsto (\varphi, \varphi') \\ \beta_2 & : (0, 1) \mapsto (\chi, \chi') \end{aligned}$$

These bases correspond to the following EPR-pair states:

$$\begin{aligned} |\Omega_0\rangle & = \frac{1}{\sqrt{2}} \cdot (|\vartheta\rangle_1 |\vartheta'\rangle_2 - |\vartheta'\rangle_1 |\vartheta\rangle_2) \\ |\Omega_1\rangle & = \frac{1}{\sqrt{2}} \cdot (|\varphi\rangle_1 |\varphi'\rangle_2 - |\varphi'\rangle_1 |\varphi\rangle_2) \\ |\Omega_2\rangle & = \frac{1}{\sqrt{2}} \cdot (|\chi\rangle_1 |\chi'\rangle_2 - |\chi'\rangle_1 |\chi\rangle_2) \end{aligned}$$

In order to decode the resulting pairs, the following measurement operators have to be used (see [3] for details):

$$\mathcal{M}_0 = |\vartheta\rangle\langle\vartheta|$$

$$\mathcal{M}_1 = |\varphi\rangle\langle\varphi|$$

$$\mathcal{M}_2 = |\chi\rangle\langle\chi|$$

The EPR protocol is summarized in Figure 2.5; the version of the protocol in the figure is based on the explanations given in [1].

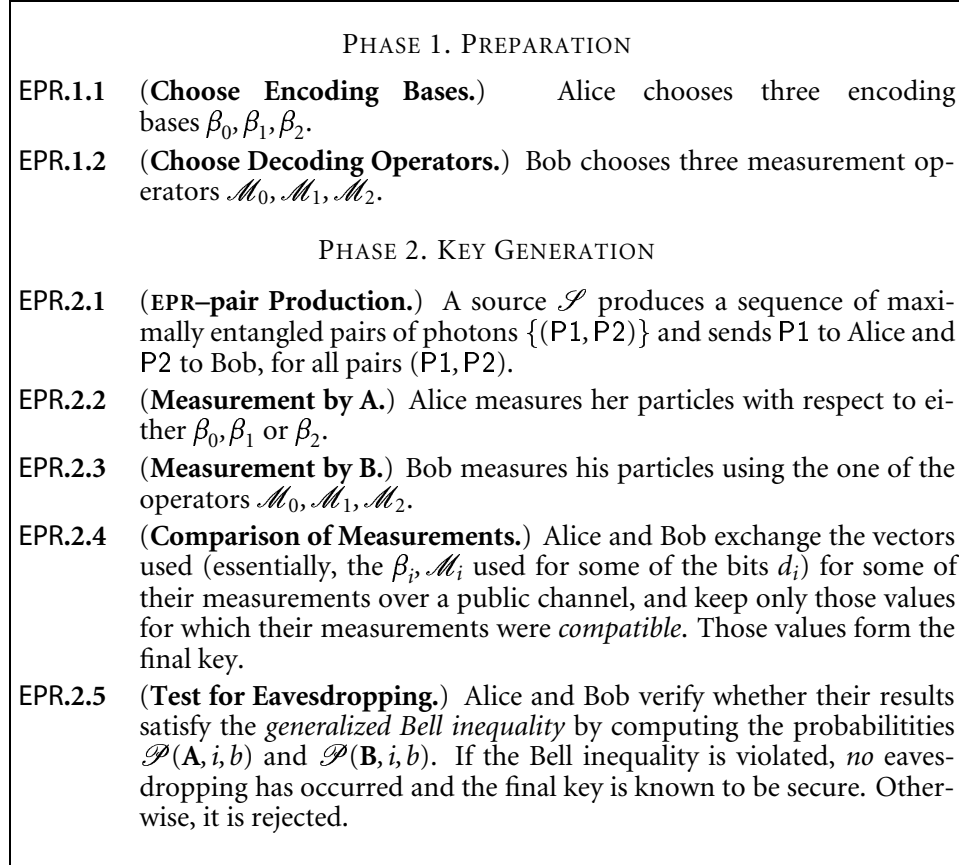


FIGURE 2.5. PROTOCOL EPR (QUANTUM KEY DISTRIBUTION).

An eavesdropper, Eve, can typically perform the following two attacks on the EPR protocol:

- Eve measures P1, P2 or both as they are transmitted from the source  $\mathcal{S}$  to Alice and Bob. In this case, the final keys in step EPR.2.4 are not the same for Alice and Bob.
- Eve substitutes the source's particles with other particles prepared by herself in advance.

Eavesdropping is detected in step EPR.2.5, which involves testing whether the Bell inequality holds. It must be noted that  $\mathcal{P}(\mathbf{A}, i, b)$  and  $\mathcal{P}(\mathbf{B}, i, b)$  are the probabilities that Alice (and Bob, respectively) obtain the bit value  $b$  when measuring with respect to  $\beta_i$  (or  $\mathcal{M}_i$ , respectively).

A formal listing of the steps in EPR is shown in Figure 2.6.

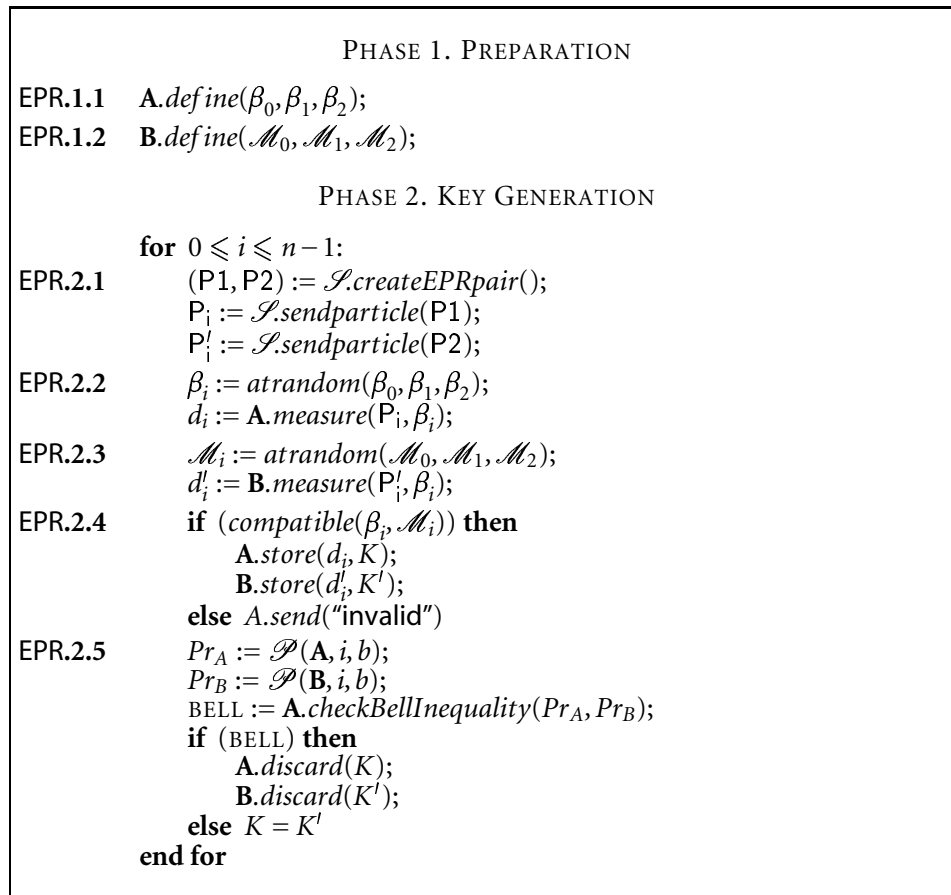


FIGURE 2.6. FORMAL DESCRIPTION OF EPR. A description of the EPR protocol in QKD-FORM, a formal, programming language – like notation.

## 2.4. Summary and Conclusion

This chapter has detailed the three quantum cryptographic protocols BB84, B92 and EPR. For each protocol, a general description was given as well as a systematic listing of the steps involved. The protocols were also expressed in QKD-FORM, providing a more-or-less complete and unambiguous description of each. This lays the foundations for Part 2 of this thesis, where the BB84 protocol will be subjected to computer modelling and validation.

## Chapter 2 Bibliography

- [1] J. Gruska. *Quantum Computing*. McGraw-Hill International, 1999.
- [2] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, third edition, 1998.
- [3] S. J. Lomonaco, Jr. A quick glance at quantum cryptography. *quant-ph/9811056*, 1998.
- [4] S. J. Lomonaco, Jr. A talk on quantum cryptography or how alice outwits eve. In D. Joyner, editor, *Coding Theory and Cryptography*. Springer, 1999.
- [5] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, first edition, 2002.



PART 2

**Modelling and Analysis of the BB84  
Protocol**



## CHAPTER 3

# A Model of BB84 in CCS

*A seal is only as good as the man in whose briefcase it's carried.*  
— KAREN SPÄRCK JONES

*In God we Trust.  
All Others we Monitor.*  
— NSA MOTTO

OF ALL THE PROTOCOLS presented in the previous chapter, it seems that B92 is the simplest: it involves the use of only one encoding basis and does not require the generation of entangled photon pairs. However, we will deal solely with the BB84 protocol in the remainder of this thesis. There are two reasons for this. The first is historical: BB84 is the oldest and the most typical quantum protocol. The other reason is that existing implementations of quantum key distribution use BB84, with very few exceptions.

As stated before, our goal will be to develop a model of BB84 and to use verification software to check whether the model satisfies necessary properties. We will assume an error-free channel, so error correction will not be considered. The property of most concern to us is *the immunity of BB84 against eavesdropping*. As stated in [3],

“Proofs of unconditional security of the BB84 protocol exist and we have no reason to doubt their correctness. Nevertheless, we argue that the modelling/analysis approach has merit for the study of this and other quantum security protocols.”

The vehicle of investigation will be the concurrency formalism CCS and the software tool CWB-NC, described briefly in Section 1.3. The reasons for choosing CCS are threefold:

- (1) it is suitable for modelling interactions within complex systems, such as security protocols;
- (2) it is capable of dealing with non-deterministic behaviour, such as that exhibited by quantum protocols;
- (3) there exist inference rules and automated verification tools for proving that a system model operates according to its specification.

Clearly CCS is appropriate for verifying protocol security.

### 3.1. Brief Review of CCS

It would take an entire book to completely present the CCS notation and all its expressive power; for a good treatment of CCS, the reader is referred to [1, 2]. What follows is a cursory review of the key concepts of CCS.

A system is modelled in CCS as a set of interacting *agents* or *processes*. These agents coexist and operate simultaneously. Furthermore, they interact through *handshakes* or *synchronizations*. Agents are typically represented by capital letters or strings starting with a capital letter (e.g. **A** for Alice, **B** for Bob, but  $\mathcal{S}$  for an EPR source, **Empty** for an empty communication channel).

Agents are capable of performing internal *actions*, denoted by “names”, i.e. lowercase strings, e.g. *perform*, *correcterrors*. These actions are performed in some particular order, which is given in the definition of the agent.

*Interactions* are a special brand of actions; they allow agents to perform certain things in tandem. To explain how this works, consider two agents **P**, **Q** which contain the actions *put* and  $\overline{put}$  respectively. When **P** and **Q** execute concurrently, these two actions will have to be performed at the same time. If one agent is busy performing another action, then the second agent has to wait in order for the synchronization to occur.

Actions and interactions can take *arguments* — in which case the “value-passing calculus” is employed. An example of a value-passing action is  $put(d, b)$ , which could denote the placement of the values *d* and *b* into part of an agent.

Agents are combined together in CCS using operators. In order for two agents to execute concurrently, they have to be *composed* using the “|” operator. Agents with actions occurring in a particular order are defined using the sequencing “.” operator. For example,  $a.A.0$  is the agent that first performs action *a*, then performs agent *A*, then stops (**0** is the so-called *inactive agent*). The “+” operator combines agents non-deterministically:  $(A + B)$  is the agent that either behaves as agent *A* or as agent *B* at a given moment.

Synchronizations in a system of concurrent agents are presented using the relabelling operator “ $\backslash \mathcal{L}$ ”. For example,  $(\mathbf{P} \mid \mathbf{Q}) \backslash \mathcal{L}$  where  $\mathcal{L} = \{get, put\}$  is the system consisting of agents **P** and **Q** which synchronize on the interactions *get* and *put*.

Finally, agents can be proved to be *equivalent*. There are various kinds of equivalence, including behavioural equivalence, congruence, bisimulation, trace equivalence, all defined meticulously in [1]. In our effort to prove the security of BB84 we will use only the notion of trace equivalence.

### 3.2. The BB84 Protocol Model

The BB84 protocol involves two main parties: Alice (**A**) and Bob (**B**). These parties will be modelled as two distinct CCS agents. Alice and Bob interact by means of a quantum communication channel (the public channel is unnecessary in a basic protocol model). The channel is modelled by two CCS agents, **Empty** and  $\mathbf{Full}(d, b)$ , denoting an empty channel and a channel containing a single data bit *d* encoded as a photon using the basis *b*.

The aforementioned agents are defined as follows:

$$\begin{aligned}
 \mathbf{A} &\stackrel{\text{def}}{=} \text{choose}(x). (\overline{put}(x, 0). \overline{reveal}(0). \mathbf{go.A} + \overline{put}(x, 1). \overline{reveal}(1). \mathbf{go.A}) \\
 \mathbf{B} &\stackrel{\text{def}}{=} \overline{measure}(0). \text{get}(x). (\text{reveal}(b). \text{if } b = 0 \text{ then } \overline{keep}(x). \overline{g0}. \mathbf{B} \text{ else } \overline{g0}. \mathbf{B}) + \\
 &\quad + \overline{measure}(1). \text{get}(x). (\text{reveal}(b). \text{if } b = 1 \text{ then } \overline{keep}(x). \overline{g0}. \mathbf{B} \text{ else } \overline{g0}. \mathbf{B}) \\
 \mathbf{Empty} &\stackrel{\text{def}}{=} \text{put}(d, b). \mathbf{Full}(d, b) \\
 \mathbf{Full}(d, b) &\stackrel{\text{def}}{=} \overline{measure}(b'). \text{if } b' = b \text{ then } \overline{get}(d). \mathbf{Empty} \\
 &\quad \text{else } (\overline{get}(0). \mathbf{Empty} + \overline{get}(1). \mathbf{Empty})
 \end{aligned}$$

The behaviour of these agents is illustrated below in the form of state transition diagrams.

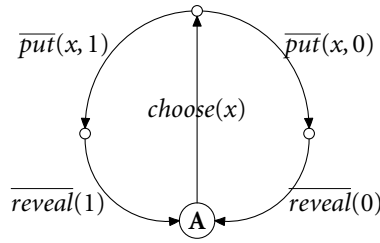


FIGURE 3.1. The agent **A** representing Alice.

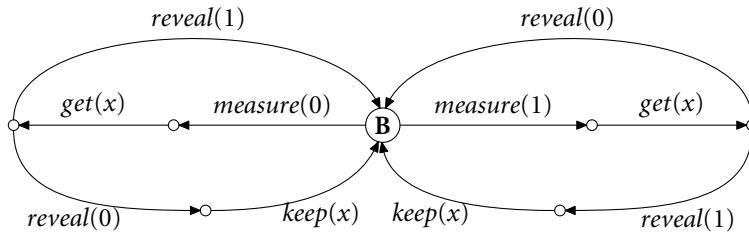


FIGURE 3.2. The agent **B** representing Bob.

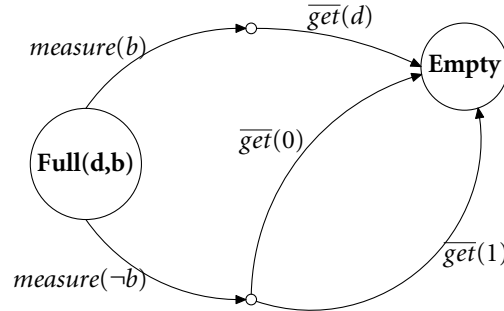


FIGURE 3.3. The interaction of the two channel agents.

### 3.2.1. Case of No Eavesdropping

Assuming that no eavesdropping occurs, the basic model of BB84 consists of the composition of Alice, Bob and the channel. The protocol model in this case can be written as the agent (note the absence of the **Full**(*d, b*) agent, which is contained within *Empty*):

$$(3) \quad \mathbf{BB84} = (\mathbf{A} \mid \mathbf{B} \mid \mathbf{Empty}) \setminus \{put, get, measure, go, reveal\}$$

As indicated in (3), the actions *put, get, measure, go, reveal* are synchronizations. Any agent needing to write to or read from the quantum channel (i.e. perform the *put, get, measure* actions) needs to synchronize with the channel (meaning that the channel needs to be ready). The *go* synchronization is necessary so as to ensure that Alice does not place values on the channel before Bob has processed them. When a measurement is revealed, both Alice and Bob synchronize via the *reveal* synchronization.

### 3.2.2. Case of Eavesdropping

If an eavesdropper is present, then she needs to be modelled as a separate agent:

$$\mathbf{E} \stackrel{\text{def}}{=} \overline{\text{measure}}(0).\text{get}(x).\overline{\text{put}}(x,0).\mathbf{E} + \overline{\text{measure}}(1).\text{get}(x).\overline{\text{put}}(x,1).\mathbf{E}$$

This assumes that the eavesdropper performs what is known as an *intercept–resend* attack. This attack can be explained directly in terms of the definition of  $\mathbf{E}$ : if she intercepts (measures) a 0 on the channel, she places a 0 on the channel again; if she intercepts a 1, she places a 1 on the channel again.

Another possible eavesdropping attack would be to place a random value on the channel after measuring it. For such an attack, the eavesdropper must be modelled as follows:

$$\mathbf{E} \stackrel{\text{def}}{=} \overline{\text{measure}}(0).\text{get}(x).(\overline{\text{put}}(x,0) + \overline{\text{put}}(x,1)).\mathbf{E} + \\ + \overline{\text{measure}}(1).\text{get}(x).(\overline{\text{put}}(x,0) + \overline{\text{put}}(x,1)).\mathbf{E}$$

However, we will assume an intercept–resend attack.

The complete protocol when the eavesdropper is taken into account can be written:

$$(4) \quad \mathbf{BB84}' = (\mathbf{A} \mid \mathbf{B} \mid \mathbf{E} \mid \mathbf{Empty}) \setminus \{\text{put}, \text{get}, \text{measure}, \text{go}, \text{reveal}\}$$

\*\*\*

### 3.2.3. Desired Behaviour: The Specification of BB84

Amongst the actions in the agents  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{E}$ ,  $\mathbf{Empty}$ ,  $\mathbf{Full}(d,b)$ , only *choose*( $x$ ) and *keep*( $x$ ) are internal. In other words, all other actions are synchronizations. The action *choose*( $x$ ) denotes the possibility that Alice chooses to send the value  $x$  into the channel. The action *keep*( $x$ ) denotes the case in which a received bit  $x$  is kept by both Alice and Bob, and forms part of the final key.

The desired behaviour of the protocol can be expressed as follows:

“For each bit  $x$  chosen by Alice, if a different basis is used by Bob for decoding,  $x$  is discarded; otherwise  $x$  is kept.”

This property can be expressed as a CCS agent:

$$(5) \quad \mathbf{Spec} = \text{choose}(x).(\mathbf{Spec} + \overline{\text{keep}}(x).\mathbf{Spec})$$

This completes the CCS model of BB84 protocol.

\*\*\*

### 3.2.4. Verification of BB84 using CWB–NC

In order to check whether BB84 operates according to the specification given previously, it is necessary to compare the agents given by equations (3), (4) and (5). To state this more concretely, we need to find out whether:

$$(6) \quad \mathbf{BB84} = \mathbf{Spec}$$

$$(7) \quad \mathbf{BB84}' = \mathbf{Spec}$$

There are two possible approaches to solving this problem:

- The inference rules of CCS can be applied manually, in order to prove whether the above agents are equivalent. The relations for the various kinds of agent equivalence then need to be used.
- An automated verification tool for CCS agents can be used to check this directly.

The second approach is more direct and generic, and it is that approach that we have taken. The CWB–NC software tool has been used for this purpose.

### 3.3. Results

The CWB–NC proves that (6) holds, but (7) does not. In particular, the case in which eavesdropping does not occur (agent **BB84**) is *trace-equivalent* to the specification. Two agents are said to have trace equivalence if their sets of possible, observable actions are identical (i.e. they have the same *trace*).

This result is precisely what is required; when there is no eavesdropper, a key is securely established between Alice and Bob. To make this clear, the trace of agent **BB84** is:

$$\text{choose}(x). (\text{Spec} + \overline{\text{keep}}(x).\text{Spec})$$

which is exactly the same as agent **Spec**. This means that, all the bits  $x$  that Alice chooses are indeed kept at the end of the protocol, forming part of the final key.

Conversely, when an eavesdropper *is* present, the sequence of actions:

$$\text{choose}(0).\overline{\text{keep}}(1)$$

arises in the trace of agent **BB84'**. This corresponds to the situation in which the value received by Bob is different from that originally sent by Alice. In terms of the underlying physics, this can only happen if the quantum channel has been observed by a third party or if an error has arisen on the channel — both these occasions would perturb the value originally sent. Since an error-free channel is assumed, the former case is the only explanation. Thus, theory agrees with experiment, and the trace shown above is caused by eavesdropping.

Essentially, it has been shown that, indeed, if eavesdropping occurs during BB84 a key will not be established. To rephrase this argument, BB84 is, indeed, perfectly capable of allowing two parties to establish a secure cryptographic key.

### A Note Regarding the Use of CWB–NC

The definitions of agents given in the previous sections cannot be handled *per se* by the CWB–NC tool. The reason is that they have been expressed in the value-passing calculus, which this tool does not support. Furthermore, the conditional statements (**if, then, else**) are notational conveniences which are not part of proper CCS. In order to perform the verification described above, the VP utility, written by Glenn Bruns<sup>1</sup>

### 3.4. Summary and Conclusion

In this chapter, a CCS model of BB84 for the cases with and without eavesdropping has been presented. The model consists of a total of five agents, one for each

<sup>1</sup>VP is available for download from <http://www.bell-labs.com/user/grb/vp/vp.html>. VP converts an input file written in the value-passing calculus into a file compatible with CWB–NC. The author of VP has kindly supplied a version specially compiled for Linux.

involved party (Alice, Bob, and the eavesdropper), and two for the communication channel. It has been shown that BB84 permits the establishment of a secret cryptographic key in the absence of eavesdropping. If eavesdropping occurs, it turns out that no key can be established. Clearly, Alice and Bob can restart the protocol until it succeeds, in this latter case.

### Chapter 3 Bibliography

- [1] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
- [2] R. Milner. *Communicating and Mobile Systems: The  $\pi$ -Calculus*. Cambridge University Press, 2001.
- [3] R. Nagarajan and S. Gay. Formal verification of quantum protocols. [quant-ph/0203086](#), March 2002.

## Analysis of BB84 using PRISM

SINCE CCS CAN ONLY MODEL what *could* occur in BB84 when all events are equally possible, a more powerful language needs to be used to fully describe the behaviour of the protocol. Furthermore, a tool is required that can accept the model of the protocol and verify that it satisfies specific properties — in particular, that the protocol is immune against eavesdropping. CWB-NC allows the user to verify that two agents are equivalent, and thus the equivalence of a system to its specification can be checked. The additional functionality needed, though, is specifically:

- the model should incorporate the *probability* with which things happen during execution of the protocol, such as the probability with which Bob detects a particular bit correctly on the channel, or the probability that an eavesdropper goes completely unnoticed.
- the model should be *verifiable*, i.e. there should be an automated way of checking that the model satisfies a given number of properties, and if so, with what probability.

This functionality is catered for by *probabilistic model checkers*. PRISM is a probabilistic model checker, as described in section 1.3.1. The primary references for PRISM are [2, 4].

The objective of this chapter is to discuss the use of the PRISM tool for modelling the BB84 quantum cryptographic protocol and verifying that it is secure against eavesdropping for a particular number of bits.

### 4.1. A PRISM Model of BB84

A PRISM model of BB84 has been constructed<sup>1</sup>. Two versions of the protocol were considered, one in which all bits of the cryptographic key are sent *together* over the quantum channel (the “multiple-bit version”), and one in which one bit is sent over the channel at a time (the “single-bit version”).

#### 4.1.1. The Multiple-Bit Version

In the model of the multiple-bit version of BB84, exactly 5 bits are transmitted together over the quantum channel (as photons). What follows is a walkthrough of the code for this model.

**4.1.1.1. Initial Comments and Declarations.** The PRISM code for BB84 is divided into five sections. The first section consists of comments and a declaration of the type of the model, which is **nondeterministic**. A nondeterministic model

---

<sup>1</sup>The code for this model is due to Dr. David Parker, Dr. Gethin Norman, and Dr. Rajagopal Nagarajan, with whom the author has collaborated.

is suitable in this case, because what is being described is made up of four probabilistic systems operating in parallel: Alice, Bob, Eve (the eavesdropper) and the quantum channel.

Next, the probabilities assumed in the model are declared. The variable **LUCKY** denotes the probability of retrieving a correct value from the channel when the incorrect basis is used. The variable **UNLUCKY** denotes precisely the opposite, i.e. the probability of obtaining an incorrect value with an incorrect basis.

Then three predicates are defined: `detected`, `notdetected`, `dontknow`. These predicates are used in order to determine whether eavesdropping has occurred. The first predicate holds when eavesdropping is detected; the second predicate holds when eavesdropping goes unnoticed. The predicate `dontknow` is true when Bob cannot be sure whether eavesdropping has occurred or not.

In the definitions of these predicates, several variables are compared together; these variables represent bit values and bases belonging to Alice and Bob. In particular:

- The variables starting with `ad` are data values belonging to Alice;
- The variables starting with `ab` are encoding bases chosen by Alice;
- The variables starting with `bd` are data values belonging to Bob;
- The variables starting with `bb` are decoding bases chosen by Bob.

The code for the aforementioned declarations follows.

---

```

1 // BB84 - Quantum Key Distribution
2 // dxp/gxn 12/2/03 and NP 16/4/03
3
4 nondeterministic
5
6 // QUANTUM READ
7 // Probability of reading correctly with wrong basis:
8 prob LUCKY = 0.5;
9 // Probability of reading incorrectly with wrong basis
10 prob UNLUCKY = 0.5;
11
12 // DEFINITIONS used by Bob to decide what has happened:
13 formula detected = (bb1=ab1) & (bd1≠ad1) |
14                   (bb2=ab2) & (bd2≠ad2) |
15                   (bb3=ab3) & (bd3≠ad3) |
16                   (bb4=ab4) & (bd4≠ad4) |
17                   (bb5=ab5) & (bd5≠ad5);
18
19 formula notdetected = ((bb1=ab1) & (bd1=ad1) | (bb1≠ab1)) &
20                       ((bb2=ab2) & (bd2=ad2) | (bb2≠ab2)) &
21                       ((bb3=ab3) & (bd3=ad3) | (bb3≠ab3)) &
22                       ((bb4=ab4) & (bd4=ad4) | (bb4≠ab4)) &
23                       ((bb5=ab5) & (bd5=ad5) | (bb5≠ab5)) &
24                       !((bb1≠ab1) & (bb2≠ab2) & (bb3≠ab3) & (bb4≠ab4) & (bb5≠ab5));
25
26 formula dontknow = (bb1≠ab1) & (bb2≠ab2) & (bb3≠ab3) & (bb4≠ab4) & (bb5≠ab5);

```

---

**4.1.1.2. Definition of Quantum Channel.** The remainder of the PRISM model defines the various modules (agents, entities) that are involved in the BB84 protocol. The first such module is the quantum communication channel, over which the data bits are sent as photons with a given polarisation.

Each module in PRISM begins with a declaration of all variables used. There is always a variable for the local state of the module; during execution of the model, this state changes. Other variables involved in the model of the channel are the data bit, *cd*, and the basis with which it is encoded, *cb*.

Each subsequent line in the module listing is of the form:

$$[\text{sync}] \quad s = X_0 \{ \& p_1 = X_1 \& p_2 = X_2 \& \dots \} \rightarrow s' = Y_0 \{ \& q_1 = Y_1 \& q_2 = Y_2 \& \dots \}$$

Here, *sync* is a synchronization label, which is used to make sure that this line of code is executed in synchrony with lines in other modules with the same label. The next item in the line of code, the *condition*  $s = X_0$ , involves the *current state*  $s$  of the module. Each such condition is checked when the module is executed, and if it holds, the action on the right hand side of the arrow is executed. Optionally, other conditions can be checked too, and these are the conditions in the  $\{ \}$  braces. On the right hand side of the arrow, the *next state* is shown. When the condition holds, the module takes on the next state. Thus, if the current state of the module is  $s = X_0$ , then the module proceeds to the next state  $s' = Y_0$ . The other assignments on the right hand side are optional.

To make this clearer, consider line 10 of the code for the channel module (shown below). In order for this line of code to be executed, the channel process must wait for the *evemeasure* action to be initiated by the Eve module. When this happens, the condition  $cs=1$  is checked; in other words, it is determined whether the current state of the channel module,  $cs$ , is 1. If so, it is determined whether  $cb=eb$  holds, i.e. whether the channel basis is equal to the basis used by Eve. If and only if these two conditions hold, the module enters the next state, shown on the right hand side of the arrow, which is  $cs'=2$ .

The code for the quantum channel is presented below.

---

```

1 // QUANTUM COMMUNICATION CHANNEL
2 module channel
3 cs: [0..6]; // local state
4 cb: [0..1]; // basis
5 cd: [0..1]; // data
6
7     // Incoming data from Alice
8     [aliceput]   cs=0 → (cs'=1) & (cb'=ab) & (cd'=ad);
9     // Data read by Eve (in 2 steps)
10    [evemeasure] cs=1 & cb=eb → (cs'=2);
11    // Eve has right basis
12    [evemeasure] cs=1 & cb≠eb → UNLUCKY : (cs'=2) & (cd'=1-cd) +
13    LUCKY : (cs'=2) & (cd'=cd);
14    // Eve has wrong basis
15    [eveget]     cs=2 → (cs'=3);
16    // Incoming data from Eve
17    [eveput]     cs=3 → (cs'=4) & (cb'=eb) & (cd'=ed);
18    // Data read by Bob (in 2 steps)
19    // Bob has right basis
20    [bobmeasure] cs=4 & cb=bb → (cs'=5);

```

```

21 // Bob has wrong basis
22 [bobmeasure] cs=4 & cb≠bb → UNLUCKY : (cs'=5) & (cd'=1-cd) +
23 LUCKY : (cs'=5) & (cd'=cd);
24 [bobget] cs=5 → (cs'=0);
25 endmodule

```

---

**4.1.1.3. Definition of Alice.** Each party taking part in the protocol has a module representing its behaviour. First of all, Alice's behaviour is defined.

Alice holds a set of five data bits  $ad_1, \dots, ad_5$  and five corresponding encoding bases  $ab_1, \dots, ab_5$ . The values are selected at random in lines 20–23 and stored in these variables in lines 24–29. This module synchronizes with the channel in order to place data on to it.

---

```

1 // ALICE
2 module alice
3 as : [0..6]; // local state
4 ai : [1..5]; // index
5 ab : [0..1]; // basis
6 ad : [0..1]; // data
7
8 ab1 : [0..1]; // all bases
9 ab2 : [0..1];
10 ab3 : [0..1];
11 ab4 : [0..1];
12 ab5 : [0..1];
13
14 ad1 : [0..1]; // all data
15 ad2 : [0..1];
16 ad3 : [0..1];
17 ad4 : [0..1];
18 ad5 : [0..1];
19
20 // Choose basis (randomly) ( after Eve has chosen her basis)
21 [] as=0 & es>0 → 0.5:(as'=1) & (ab'=0) + 0.5:(as'=1) & (ab'=1);
22 // Choose data (randomly)
23 [] as=1 → 0.5:(as'=2) & (ad'=0) + 0.5:(as'=2) & (ad'=1);
24 // Store basis/data
25 [] as=2 & ai=1 → (as'=3) & (ab1'=ab) & (ad1'=ad);
26 [] as=2 & ai=2 → (as'=3) & (ab2'=ab) & (ad2'=ad);
27 [] as=2 & ai=3 → (as'=3) & (ab3'=ab) & (ad3'=ad);
28 [] as=2 & ai=4 → (as'=3) & (ab4'=ab) & (ad4'=ad);
29 [] as=2 & ai=5 → (as'=3) & (ab5'=ab) & (ad5'=ad);
30 // Send data to channel
31 [aliceput] as=3 → (as'=4);
32 // Loop (go onto next bit , unless finished )
33 // not finished :
34 [loop] as=4 & ai<5 → (as'=0) & ( ai '=ai+1);
35 // finished :
36 [loop] as=4 & ai=5 → (as'=5);
37 // Reveal bases/data to Bob
38 [reveal] as=5 → (as'=6);
39 // stop
40 [stop] as=6 → (as'=6);

```

41 **endmodule**

**4.1.1.4. Definition of Bob.** Bob holds a set of five data bits  $bd_1, \dots, bd_5$  and five corresponding encoding bases  $bb_1, \dots, bb_5$ . He reads encoded data off the channel, and synchronizes with Alice in order to discover how many of the read bits are correct.

---

```

1 // BOB
2 module bob
3 bs : [ 0..8 ]; // local state
4 bb : [ 0..1 ]; // basis
5 bd : [ 0..1 ]; // data
6
7 bb1 : [ 0..1 ]; // all bases
8 bb2 : [ 0..1 ];
9 bb3 : [ 0..1 ];
10 bb4 : [ 0..1 ];
11 bb5 : [ 0..1 ];
12
13 bd1 : [ 0..1 ]; // all data
14 bd2 : [ 0..1 ];
15 bd3 : [ 0..1 ];
16 bd4 : [ 0..1 ];
17 bd5 : [ 0..1 ];
18
19 // Choose basis (randomly) ( after Eve has chosen her basis)
20 [] bs=0 & es>0 → 0.5 : ( bs'=1 ) & ( bb'=0 ) + 0.5 : ( bs'=1 ) & ( bb'=1 );
21 // Read data from channel
22 [bobmeasure] bs=1 → ( bs'=2 );
23 [bobget] bs=2 → ( bs'=3 ) & ( bd'=cd );
24 // Store basis/data
25 [] bs=3 & ai=1 → ( bs'=4 ) & ( bb1'=bb ) & ( bd1'=bd );
26 [] bs=3 & ai=2 → ( bs'=4 ) & ( bb2'=bb ) & ( bd2'=bd );
27 [] bs=3 & ai=3 → ( bs'=4 ) & ( bb3'=bb ) & ( bd3'=bd );
28 [] bs=3 & ai=4 → ( bs'=4 ) & ( bb4'=bb ) & ( bd4'=bd );
29 [] bs=3 & ai=5 → ( bs'=4 ) & ( bb5'=bb ) & ( bd5'=bd );
30 // Loop (go onto next bit , unless finished )
31 // not finished :
32 [loop] bs=4 & ai<5 → ( bs'=0 );
33 // finished :
34 [loop] bs=4 & ai=5 → ( bs'=5 );
35 // See Alice 's bases/data
36 // all different bases
37 [reveal] bs=5 & dontknow → ( bs'=6 );
38 // where bases same, data same
39 [reveal] bs=5 & notdetected → ( bs'=7 );
40 // some bases same but data different
41 [reveal] bs=5 & detected → ( bs'=8 );
42 // Stop - don't know anything
43 [stop] bs=6 → ( bs'=6 );
44 // Stop - nothing detected
45 [stop] bs=7 → ( bs'=7 );
46 // Stop - detected eavesdropper

```

```

47     [stop] bs=8 → (bs'=8);
48
49 endmodule

```

---

**4.1.1.5. Definition of Eve.** Eve reads data off the channel and puts the result of her measurement back on to it. This is the so-called intercept–resend attack.

---

```

1 // EVE
2 module eve
3 es : [ 0..4 ]; // local state
4 eb : [ 0..1 ]; // basis
5 ed : [ 0..1 ]; // data
6
7     // choose basis ( nondeterministically )
8     [] es=0 → (es'=1) & (eb'=0);
9     [] es=0 → (es'=1) & (eb'=1);
10    // read data from channel
11    [evemeasure] es=1 → (es'=2);
12    [eveget] es=2 → (es'=3) & (ed'=cd);
13    // send data to channel
14    [eveput] es=3 → (es'=0);
15 endmodule

```

---

#### 4.1.2. The Single–Bit Version

The code for the single-bit version is listed in Appendix A. This code does not differ greatly from the multiple-bit version, the most important change being the line “**const** N = 5;”. This line can be changed in order to model the execution of BB84 with an arbitrary number of transmitted bits. QKD systems with arbitrary key lengths can thus be described.

### 4.2. Desired Properties of the Protocol

The goal of BB84 is to share a secret cryptographic key, and the laws of quantum mechanics are used to ensure that this key is untampered with by a third party. This is precisely the property that must be verified. Therefore, it is necessary to establish that BB84 is immune against eavesdropping.

As far as the PRISM models are concerned, the probability of detecting an eavesdropper needs to be computed; the greater this probability, the more the certainty that the key is secure. In the design of a practical QKD system, this probability needs to be maximized.

The probabilities that need to be computed for the models developed previously are:

- the probability of detecting an eavesdropping attack,  $\mathcal{P}_{det}$ ;
- the probability of not detecting an eavesdropping attack,  $\mathcal{P}_{nodet}$ ;
- the probability with which neither of the above occurs,  $\mathcal{P}_{dn}$ .

The following listing is the properties file for the multiple-bit version. Note that PRISM computes lower bounds (those cases defined by  $P \geq 0.5$ ) and upper bounds (those cases defined by  $P \leq 0.5$ ) on the probabilities.

---

```

1 // PROPERTIES FOR MULTIPLE-BIT VERSION
2 // Lower Bound Probabilities :
3
4 // prob. don't know anything
5  $P \geq 0.5$  [ true U bs=6 {cs=0&as=0&ai=1&bs=0&es=0} ]
6 // prob. nothing detected
7  $P \geq 0.5$  [ true U bs=7 {cs=0&as=0&ai=1&bs=0&es=0} ]
8 // prob. detected eavesdropper
9  $P \geq 0.5$  [ true U bs=8 {cs=0&as=0&ai=1&bs=0&es=0} ]
10
11 // Upper Bound Probabilities :
12
13 // prob. don't know anything
14  $P \leq 0.5$  [ true U bs=6 {cs=0&as=0&ai=1&bs=0&es=0} ]
15 // prob. nothing detected
16  $P \leq 0.5$  [ true U bs=7 {cs=0&as=0&ai=1&bs=0&es=0} ]
17 // prob. detected eavesdropper
18  $P \leq 0.5$  [ true U bs=8 {cs=0&as=0&ai=1&bs=0&es=0} ]

```

---

The corresponding properties file for the single-bit version is listed next.

```

1 // PROPERTIES FOR SINGLE-BIT VERSION
2 // Lower Bound Probabilities :
3
4 // prob. don't know anything
5  $P \geq 0.5$  [ true U bs=5 {cs=0&as=0&ai=1&bs=0&es=0} ]
6 // prob. nothing detected
7  $P \geq 0.5$  [ true U bs=6 {cs=0&as=0&ai=1&bs=0&es=0} ]
8 // prob. detected eavesdropper
9  $P \geq 0.5$  [ true U bs=7 {cs=0&as=0&ai=1&bs=0&es=0} ]
10
11 // Upper Bound Probabilities :
12
13 // prob. don't know anything
14  $P \leq 0.5$  [ true U bs=5 {cs=0&as=0&ai=1&bs=0&es=0} ]
15 // prob. nothing detected
16  $P \leq 0.5$  [ true U bs=6 {cs=0&as=0&ai=1&bs=0&es=0} ]
17 // prob. detected eavesdropper
18  $P \leq 0.5$  [ true U bs=7 {cs=0&as=0&ai=1&bs=0&es=0} ]

```

---

We shall use the following notation for the probabilities with which the above properties are satisfied:

Symbol	Meaning
$\mathcal{P}_{det}^+$	Upper bound for probability of detecting eavesdropping
$\mathcal{P}_{det}^-$	Lower bound for probability of detecting eavesdropping
$\mathcal{P}_{nodet}^+$	Upper bound for probability of not detecting eavesdropping
$\mathcal{P}_{nodet}^-$	Lower bound for probability of not detecting eavesdropping
$\mathcal{P}_{dn}^+$	Upper bound for probability of other cases
$\mathcal{P}_{dn}^-$	Lower bound for probability of other cases

Using this convention, we can write:

$$(8) \quad \mathcal{P}_{det}^- \leq \mathcal{P}_{det} \leq \mathcal{P}_{det}^+$$

$$(9) \quad \mathcal{P}_{nodet}^- \leq \mathcal{P}_{nodet} \leq \mathcal{P}_{nodet}^+$$

$$(10) \quad \mathcal{P}_{dn}^- \leq \mathcal{P}_{dn} \leq \mathcal{P}_{dn}^+$$

Also, the probabilities cover all possibilities, so sum together to 1:

$$(11) \quad \sum \mathcal{P} = \mathcal{P}_{det} + \mathcal{P}_{nodet} + \mathcal{P}_{dn} = 1$$

It is expected that, if more bits are used,  $P_{dn} \rightarrow 0$  and  $P_{det} \rightarrow 1$ . Also, the probability of not detecting eavesdropping must be minimized. In more mathematical terms, the desirable behaviour can be expressed as:

$$(12) \quad \lim_{N \rightarrow \infty} P_{dn}(N) = 0$$

$$(13) \quad \lim_{N \rightarrow \infty} P_{nodet}(N) = 0$$

$$(14) \quad \lim_{N \rightarrow \infty} P_{det}(N) = 1$$

This is shown to be true for the single-bit version of the protocol in section 4.3.3. The PRISM tool is now used to obtain the values of the above probabilities.

### 4.3. Verification of the PRISM Model

When PRISM is supplied with the input files presented in the previous sections, it computes all possible states of the models and the probabilities with which the given properties hold.

#### 4.3.1. Property-Checking Timings

The multiple-bit version takes a long time to verify, even for only 5 bits of data (see Figure 4.1). For this reason, it is computationally expensive to calculate the probabilities for more bits. The single-bit version is more amenable to calculation, and the time required for PRISM to obtain the probabilities is shown in Figure 4.2. All measurements were made on a PC with an Intel Deschutes 450MHz processor with 384MB of RAM.

Probability	Computational Time (sec)
$\mathcal{P}_{det}^+$	225.897
$\mathcal{P}_{det}^-$	182.694
$\mathcal{P}_{nodet}^+$	240.972
$\mathcal{P}_{nodet}^-$	205.948
$\mathcal{P}_{dn}^+$	38.145
$\mathcal{P}_{dn}^-$	35.317

FIGURE 4.1. Time taken to compute probabilities for multiple-bit version with  $N = 5$ .

#### 4.3.2. Results for Multiple-Bit Version

Figure 4.3, indicates the probabilities computed by PRISM for the multiple-bit version of BB84, when exactly 5 bits are transmitted. It is well worth noting that the upper and lower bounds are actually equal ( $P_{det}^+ = P_{det}^- = P_{det}$ ,  $P_{nodet}^+ = P_{nodet}^- = P_{nodet}$ ,  $P_{dn}^+ = P_{dn}^- = P_{dn}$ ). By rounding the probabilities to 5 decimal places, we find, as expected, that:

$$(15) \quad \sum \mathcal{P} = \mathcal{P}_{det} + \mathcal{P}_{nodet} + \mathcal{P}_{dn} = 0.48709 + 0.48166 + 0.03125 = 1$$

The results clearly show that this particular multiple-bit version does not behave satisfactorily. There is a non-negligible probability  $P_{dn}$ , indicating that there will be several occasions in which Bob will be unable to determine whether eavesdropping has occurred. Also, the probability of detecting eavesdropping is below 50%, which means that there is a fair chance of eavesdropping going unnoticed.

### 4.3.3. Results for Single-Bit Version

The single-bit version of the protocol, listed in Appendix A, is capable of modelling the exchange of an arbitrary number of bits, as long as they are transmitted one at a time. All that is required to change the number of bits modelled is a change in line 11, page 55.

The desired behaviour defined previously is satisfied at large by the single-bit version when  $N = 64$  bits are transmitted. The values of the probabilities in this case are shown in Figure 4.4.

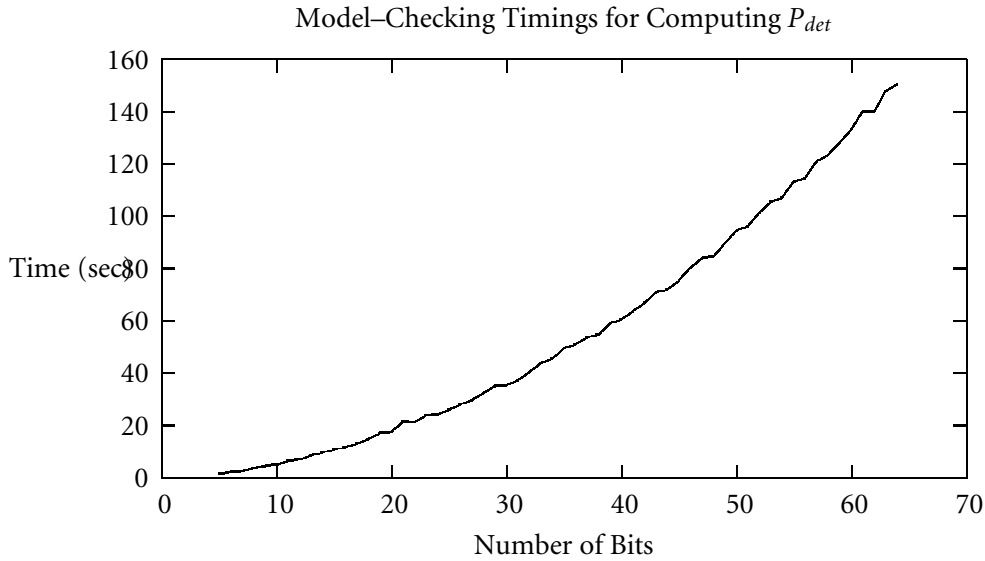


FIGURE 4.2. Time taken to compute  $P_{det}^-$  for key lengths of 5 to 64 bits.

Probability	Computed Value
$\mathcal{P}_{det}^+$	0.487091064453125
$\mathcal{P}_{det}^-$	0.487091064453125
$\mathcal{P}_{nodet}^+$	0.481658935546875
$\mathcal{P}_{nodet}^-$	0.481658935546875
$\mathcal{P}_{dn}^+$	0.03125
$\mathcal{P}_{dn}^-$	0.03125

FIGURE 4.3. Probabilities computed by PRISM for multiple-bit version of BB84 with  $N = 5$ .

Probability	Computed Value
$\mathcal{P}_{det}^+$	0.9998056809433629
$\mathcal{P}_{det}^-$	0.9998056809433629
$\mathcal{P}_{nodet}^+$	0.0019431905663704
$\mathcal{P}_{nodet}^-$	0.0019431905663704
$\mathcal{P}_{dn}^+$	0.0
$\mathcal{P}_{dn}^-$	0.0

FIGURE 4.4. Probabilities computed by PRISM for single-bit version of BB84 with  $N = 64$ .

Using UNIX shell scripts (given in Appendix C), the author was able to extract the probabilities computed by PRISM for different variations of the single-bit version. The results are plotted in Figure 4.5, and the trend is obvious: as the transmitted bits increase, the probability of detecting eavesdropping tends asymptotically towards 1. Hence, equation 14 is satisfied.

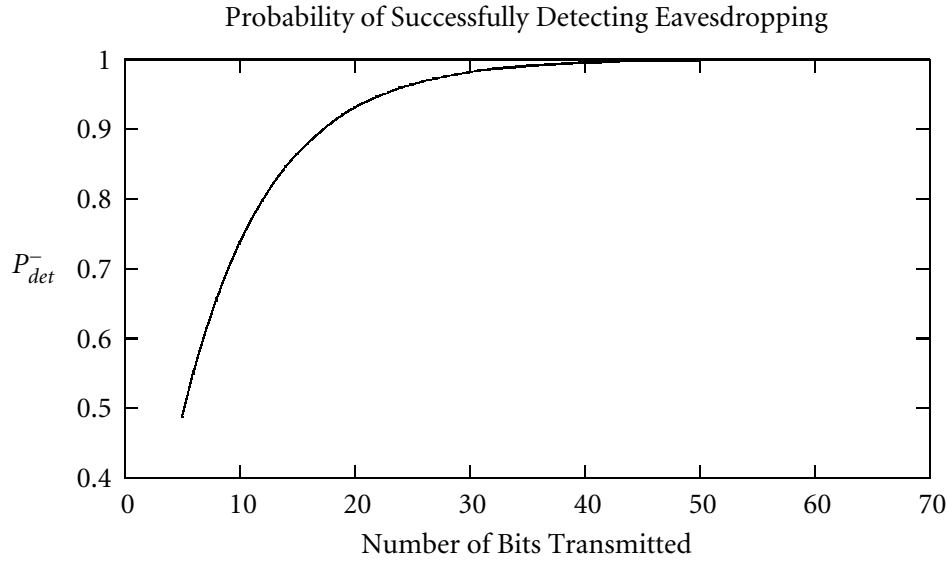


FIGURE 4.5.  $P_{det}^-$  versus  $N$ , where  $5 \leq N \leq 64$  bits.

Similarly, equations (13) and (12) are satisfied by the model. This can be seen in Figures 4.6 and 4.7.

#### 4.4. Relation of Results to Theory

It is necessary to compare the results that have been obtained with PRISM with the existing theoretical results concerning BB84. According to [3], the criterion with which a QKD protocol is judged to be secure is :

- **Definition 4.1 (Secure QKD protocol).** A QKD protocol is defined as being secure if, for any security parameters  $s > 0$  and  $\lambda > 0$  chosen by Alice and Bob, and for any eavesdropping strategy, either the scheme aborts, or it succeeds with probability at least  $1 - \mathcal{O}(2^{-s})$  and guarantees that Eve's mutual information with the final key is less than  $2^{-\lambda}$ . The key string must also be essentially random.

This gives a lower bound on the probability of successfully exchanging a secret key using BB84. However, the aforementioned bound is dependent on the security

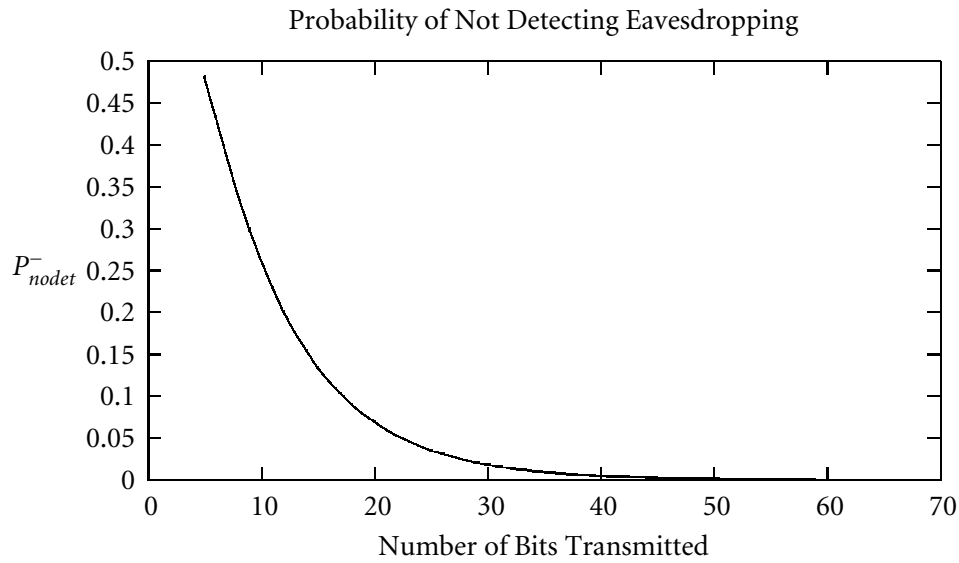
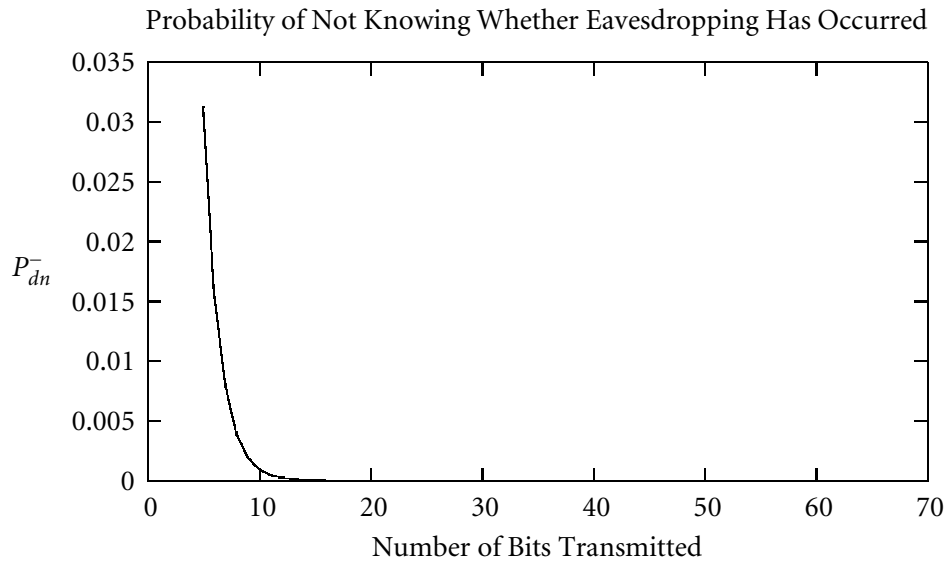
(a)  $P_{nodet}$  versus  $N$ , where  $5 \leq N \leq 64$  bits.(b)  $P_{dn}$  versus  $N$ , where  $5 \leq N \leq 64$  bits.

FIGURE 4.6. *Probability of Not Detecting Eavesdropping (a) and Probability of Not Knowing Whether Eavesdropping has Occurred (b) versus number of bits transmitted.*

parameter  $s$ , which in the models presented here is identically zero. The reader is reminded that the security parameter determines how many random subsets of the reconciled key are chosen by Alice and Bob to form the final key, and in the PRISM

code no such selection is made. Rather, the whole of the reconciled key is taken as the final key.

The probability of a successful key exchange using BB84 has been found to be [3, 1]:

$$(16) \quad \mathcal{P}_{success} = 1 - \mathbf{O}(2^{N-\lambda-s})$$

In order to obtain an analytic expression for  $\mathcal{P}_{det}$ , so it can be compared to the curve in Figure 4.5, we proceed as follows:

- The probability of success of BB84 and the probability of failure of BB84 sum to 1, according to the laws of probability:

$$\mathcal{P}_{success} + \mathcal{P}_{failure} = 1$$

- The probability of failure of BB84 is the sum of:
  - \* the probability of eavesdropping occurring and not being detected,  $\mathcal{P}_{nodet}$ , and
  - \* the probability of not being able to determine whether eavesdropping has occurred,  $\mathcal{P}_{dn}$ .
- The probability of success of BB84 is given by Equation (16).
- Hence,

$$\begin{aligned} P_{failure} &= \mathbf{O}(2^{N-\lambda-s}) \\ P_{failure} &= \mathcal{P}_{nodet} + \mathcal{P}_{dn} \\ \sum \mathcal{P} &= \mathcal{P}_{det} + \mathcal{P}_{nodet} + \mathcal{P}_{dn} \end{aligned}$$

- By combining the above expressions we obtain,

$$(17) \quad \mathcal{P}_{det} = 1 - \mathbf{O}(2^{N-\lambda-s}) \approx 1 - \mathbf{O}(2^N) \quad (\text{for small } s \text{ and } \lambda)$$

The curve of Figure 5 has the shape of a negative binary exponential function, exactly corresponding to the expression  $1 - 2^N$ . Clearly the results obtained using PRISM are in accordance with this theoretical result.

## 4.5. Summary and Conclusion

In this chapter, the requirements for modelling BB84 fully were described. Since the protocol exhibits probabilistic behaviour, a suitable model-checker needs to be used to verify whether it is indeed secure against eavesdropping. A PRISM model of BB84 in the case where 5 bits are transmitted simultaneously has been detailed. The properties that the protocol should satisfy were explained. The probability of detecting an eavesdropper during execution of BB84 for key lengths of 5 to 64 bits is computed for the case when one bit is sent at a time. Finally, the results obtained using PRISM were compared to the predictions of the relevant theory and were found to match.

It can be concluded from the results presented here that BB84 is indeed a secure key distribution protocol, and increasingly so the more the key bits that are exchanged. Furthermore, a total of 64 bits is enough to eliminate any uncertainty about whether eavesdropping has occurred at all (i.e. for  $N = 64$ ,  $\mathcal{P}_{dn} = 0$ ).

## Chapter 4 Bibliography

- [1] J. Gruska. *Quantum Computing*. McGraw-Hill International, 1999.
- [2] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic Symbolic Model Checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Computer Performance Evaluation (TOOLS'02)*, pages 200–204. Springer-Verlag, 2002.
- [3] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, first edition, 2002.
- [4] D. Parker. PRISM user's guide, September 2001. Available online at <http://www.cs.bham.ac.uk/~dxp/prism>.



PART 3

## **Epilogue**



## QKD-FORM: A Formalism for Describing Quantum Protocols

*Computer Scientists of different persuasions might put forward different arguments in favour of having a precise and accurate definition of a given programming language. Their differing backgrounds, attitudes and needs will lead to different sets of requirements and demands. But they will all be united in the view that a precise and accurate definition is desirable, if not essential.*

— ANDREW M. MCGETTRICK,  
*The Definition of Programming Languages*

IN CHAPTER 2, BB84, B92 and EPR were described in words, in a numbered summary form (e.g. Figure 2.1) and also in the formal notation QKD-FORM. This notation has been devised by the author solely for describing these protocols, but is easily extended to cater for new features.

### 5.1. Formal Definition of the QKD-FORM Syntax

Below, an EBNF grammar of the QKD-FORM syntax is given. Such a grammar could be input to a parser generator in order to create an interpreter for this notation. The primitives available are listed here as procedure names ( $\langle \text{procname} \rangle$ ); their function is explained in the next section.

$\langle \text{qkd-listing} \rangle ::= \langle \text{statement-list} \rangle$	—Definition of a QKD-FORM listing..
$\langle \text{statement} \rangle ::= \langle \text{ident} \rangle \text{' := ' } \langle \text{expr} \rangle$	—Assignment Statement.
$\quad   \langle \text{agent} \rangle \text{' : ' } \langle \text{procedure} \rangle \text{' ; '}$ $\quad   \text{' for ' } \langle \text{expr} \rangle \leq \langle \text{variable} \rangle \leq \langle \text{expr} \rangle$ $\quad \quad \langle \text{statement-list} \rangle$ $\quad \quad \text{' end for '}$	—Action performed by an agent. —Iteration.
$\quad   \text{' if ' } ( \langle \text{variable} \rangle \text{' = ' } \langle \text{expr} \rangle \text{' } ) \text{' then '}$ $\quad \quad \langle \text{statement-list} \rangle$ $\quad \quad \{ \text{' else '}$ $\quad \quad \quad \langle \text{statement-list} \rangle \}$ $\quad \quad \text{' end if '}$	—Conditional Statement.
$\langle \text{statement-list} \rangle ::= \langle \text{statement} \rangle^*$	
$\langle \text{expr} \rangle ::= [0\dots9]^*$	
$\quad   \langle \text{variable} \rangle$ $\quad   \langle \text{action} \rangle$	
$\langle \text{expr-list} \rangle ::= \langle \text{expr} \rangle \text{' , ' } \langle \text{expr-list} \rangle$	
$\langle \text{variable} \rangle ::= \langle \text{var} \rangle \{ \text{' ' } \}_{ \langle \text{optionalsubscript} \rangle }$	—Variables can optionally have a prime and/or subscript.
$\langle \text{var} \rangle ::= [a-z]^*$	
$\quad   [\alpha-\omega]^*$	

$[A-Z]^*$ $[A-Z]^*$ $[M-P]$	—These letters are reserved for quantum operators and probabilities.
$\langle \text{optionalsubscript} \rangle ::= [0..9]^*$ $[a-z]^*$	
$\langle \text{agent} \rangle ::= [A-Z]$ $[A-L]$ $[Q-Z]$	
$\langle \text{procedure} \rangle ::= \langle \text{procname} \rangle \langle ' \rangle \langle \text{expr-list} \rangle \langle ' \rangle$ $\langle \text{procname} \rangle \langle ' \rangle$	—This is for procedures without arguments.
$\langle \text{procname} \rangle ::=$ encode decode randombit randomangle randomsubsetof atrandom send sendparticle receive correcterrors measure store discard createEPRpair checkBellInequality getparity recreate	

## 5.2. The Semantics of QKD-FORM, Informally

The primitives available in QKD-FORM are operations that the parties involved in a QKD protocol might wish to produce. The primitives are:

- **Agent.encode**( $d, \beta$ ) takes a binary value  $d$  and a basis  $\beta$  (defined as a pair of angles  $(\vartheta, \varphi)$  in order to produce a photon  $P$  whose polarization corresponds to the value of  $d$  when encoded with basis  $\beta$ .
- **Agent.decode**( $P, \beta$ ) takes a photon  $P$  and a basis  $\beta$  in order to produce a binary value  $d$  corresponding to the polarization angle of  $P$ .
- **Agent.randombit**() returns 0 or 1 at random chosen with a normal distribution.
- **Agent.randomangle**() returns a random integer from 0 to 360 corresponding to an angle in degrees.
- **Agent.randomsubsetof**( $K$ ) returns a random subset  $\Xi$  of the set  $K$ , where  $K$  is a set of binary values, bases or photons.
- **Agent.atrandom**( $a_0, a_1, \dots, a_n$ ) returns one of its input arguments chosen at random with a normal distribution, where  $a_0, a_1, \dots, a_n$  can be binary values, bases or photons.
- **Agent.send**("message",  $\mathbf{B}$ ) sends an unencrypted text message over a classical communication channel to  $\mathbf{B}$ .
- **Agent.sendparticle**( $P, \mathbf{B}$ ) sends particle  $P$  over a quantum communication channel to  $\mathbf{B}$ .
- **Agent.receive**( $P$ ) denotes the event of **Agent** receiving the photon  $P$ .

- **Agent.correcterrors()** is an abstraction of any quantum error-correction procedure performed by **Agent**.
- **Agent.measure**( $P, \beta$ ) returns the result of measuring particle  $P$  using basis  $\beta$ .
- **Agent.store**( $a_0, a_1, \dots, a_n$ ) denotes the action of **Agent** storing  $a_0, a_1, \dots, a_n$  locally.
- **Agent.discard**( $a_0, a_1, \dots, a_n$ ) denotes the action of discarding  $a_0, a_1, \dots, a_n$  completely.
- $\mathcal{S}.createEPRpair()$  returns an EPR pair generated by the EPR source  $\mathcal{S}$ .
- **Agent.checkBellInequality**( $\mathcal{P}_A, \mathcal{P}_B$ ) returns 1 if the Bell Inequality is satisfied given probabilities  $\mathcal{P}_A$  and  $\mathcal{P}_B$ .
- **Agent.getparity**( $\Xi$ ) returns the parity of a set of binary values  $\Xi$ .
- **Agent.recreate**( $\Xi, I, K$ ) returns the subset  $\Xi$  of  $K$  given the set of indices  $I$  of the values in  $K$ .



## Conclusions and Future Work

*In conclusion, I would like to emphasize my belief that the era of computing chemists, when hundreds if not thousands of chemists will go to the computing machine instead of the laboratory, for increasingly many facets of chemical information, is already at hand. There is only one obstacle, namely, that someone must pay for the computing time.*

—ROBERT S. MULLIKEN (1966)

**T**HIS PROJECT HAS MANAGED to establish that BB84 is indeed a secure key distribution protocol, capable of protecting communicating parties against eavesdropping attacks. The probability of detecting an eavesdropper can be made arbitrarily close to 1, the more the bits in the key. For 64 bits, sent one at a time, the probability of a successful eavesdropping attack has been found to be just  $1.94320 \times 10^{-3}$ .

Furthermore, the computer modelling approach has proved to be flexible and powerful, albeit memory-hungry. Certainly, quantum cryptography and the protocols for QKD provide a new domain of discourse for computer scientists and engineers, definitely worthy of further work, both in theory and practice.

There are numerous directions for future work. The existing models are not complete. For example, the security parameters have not been taken into consideration in the models of BB84. Hence, there is still much to be done on the code presented in this thesis.

B92 and EPR as well as newer quantum protocols should be described in CCS and PRISM. Other modelling languages and toolsets should also be explored.

The multiple-bit version of BB84 is difficult to verify due to the large number of states that arise, even with only 5 bits. Optimization of the model for the multiple-bit version would be useful, so that a larger number of transmitted bits can be modelled.

A promising idea for future work is the creation of an operational semantics for the QKD-FORM notation and the development of a compiler that translates QKD-FORM directly to PRISM.

It is hoped that the author will be able to study these and related matters in the form of postgraduate study in the near future.

### Cumulative Project Bibliography

- [1] G. A. Barbosa, E. Corndorf, P. Kumar, H. P. Yuen, G. D' Ariano, M. Paris, and P. Perinotti. Secure communication using coherent states. [quant-ph/0210089](https://arxiv.org/abs/quant-ph/0210089), 2002.
- [2] C. H. Bennett, G. Brassard, and A. K. Ekert. Quantum cryptography. *Scientific American Special Issue: The Computer in the 21st Century*, 1995.

- [3] E. Biham, M. Boyer, P. O. Boykin, T. Mor, and V. Roychowdhury. A proof of the security of quantum key distribution. *quant-ph/9912053*, December 1999.
- [4] G. Brassard. *Modern Cryptology: A Tutorial*. LNCS 325. Springer-Verlag, 1988.
- [5] R. Cleaveland, T. Li, and S. Sims. The concurrency workbench of new century user's manual. Dept. of Computer Science, SUNY at Stony Brook, 2000.
- [6] C. Courcoubetis and S. Tripakis. Probabilistic model checking: formalisms and algorithms for discrete and real-time systems.
- [7] C. W. Dawson. *The Essence of Computing Projects: A Student's Guide*. Essence of Computing. Prentice-Hall, 2000.
- [8] P. J. Denning and R. M. Metcalfe, editors. *Beyond Calculation: The Next Fifty Years of Computing*, 1998.
- [9] A. Ekert. Quantum cryptanalysis: Introduction. *Centre for Quantum Computation (CQC)*, 2002.
- [10] A. Ekert. What is quantum cryptography? *Centre for Quantum Computation (CQC)*, 2002.
- [11] M. Gardner, editor. *Great Essays in Science*, 1985.
- [12] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden. Quantum cryptography. *Rev. Mod. Phys.*, 145(74), 2002.
- [13] A. Gough. Quantum::Entanglement. <http://www.perl.com/>, 2001.
- [14] J. Gruska. *Quantum Computing*. McGraw-Hill International, 1999.
- [15] H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Reliability. *SICS Research Report SICS/R90013*, December 1994.
- [16] G. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.
- [17] *Proceedings of the IEEE: Special Section on Cryptography*. Institute of Electrical and Electronic Engineers, May 1988.
- [18] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, third edition, 1998.
- [19] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic Symbolic Model Checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Computer Performance Evaluation (TOOLS'02)*, pages 200–204. Springer-Verlag, 2002.
- [20] S. Lipschutz. *Probability*. Schaum's Outlines. McGraw-Hill, 1965.
- [21] S. J. Lomonaco, Jr. A quick glance at quantum cryptography. *quant-ph/9811056*, 1998.
- [22] S. J. Lomonaco, Jr. A talk on quantum cryptography or how alice outwits eve. In D. Joyner, editor, *Coding Theory and Cryptography*. Springer, 1999.
- [23] J. Manners, editor. *Quantum Physics: An Introduction*. The Open University, 2000.
- [24] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
- [25] R. Milner. *Communicating and Mobile Systems: The  $\pi$ -Calculus*. Cambridge University Press, 2001.
- [26] J. Mullins. Making unbreakable code. *IEEE Spectrum*, pages 40–45, May 2002.
- [27] R. Nagarajan and S. Gay. Formal verification of quantum protocols. *quant-ph/0203086*, March 2002.
- [28] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, first edition, 2002.
- [29] D. Parker. PRISM user's guide, September 2001. Available online at <http://www.cs.bham.ac.uk/~dxp/prism>.
- [30] J. R. Pierce. *An Introduction to Information Theory: Symbols, Signals and Noise*. Dover Publications, second, revised edition, 1980.
- [31] J. Preskill. Lecture notes for physics 229: Quantum information and computation, September 1998. California Institute of Technology.
- [32] E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32:300–335, September 2000.
- [33] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.
- [34] J. Seely. *Writing Reports*. One Step Ahead. Oxford University Press, 2002.
- [35] G. Simmons. Cryptanalysis and protocol failures. *Communications of the ACM*, 37(11), 1994.
- [36] D. F. Styer. *The Strange World of Quantum Mechanics*. Cambridge University Press, 2000.
- [37] L. Takács. *Stochastic Processes: Problems and Solutions*. Methuen & Co. Ltd., 1960.
- [38] D. Welsh. *Codes and Cryptography*. Oxford Science Publications. Oxford University Press, 1998.
- [39] N. Wiener. *Cybernetics: or Control and Communication in the Animal and the Machine*. The MIT Press, second edition, 1961.
- [40] S. Wiesner. Conjugate coding. *ACM SIGACT News*, 15(1):78–88, 1983.
- [41] C. P. Williams and S. H. Clearwater. *Ultimate Zero and One*. Copernicus (Springer-Verlag), 2000.
- [42] A. Wyner. The wire-tap channel. *Bell Systems Technical Journal*, 54(8):1355–1387, 1975.

## APPENDIX A

### PRISM Model of Single-Bit Version of BB84

---

```
1 // bb84 - quantum key distribution
2 // dxp/gxn 12/2/03 and NP 16/4/03
3
4 nondeterministic
5
6 // quantum read
7 prob LUCKY = 0.5; // prob reading correctly with wrong basis
8 prob UNLUCKY = 0.5; // prob reading incorrectly with wrong basis
9
10 // number of bits to transmit
11 const N = 5;
12
13 module channel
14     cs : [ 0..6 ]; // local state
15     cb : [ 0..1 ]; // basis
16     cd : [ 0..1 ]; // data
17
18     // incoming data from alice
19     [aliceput] cs=0 → (cs'=1) & (cb'=ab) & (cd'=ad);
20     // data read by eve (in 2 steps)
21     [evemeasure] cs=1 & cb=eb → (cs'=2); // eve has right basis
22     [evemeasure] cs=1 & cb≠eb → UNLUCKY : (cs'=2) & (cd'=1-cd) +
23     LUCKY : (cs'=2) & (cd'=cd); // wrong basis
24     [eveget] cs=2 → (cs'=3);
25     // incoming data from eve
26     [eveput] cs=3 → (cs'=4) & (cb'=eb) & (cd'=ed);
27     // data read by bob (in 2 steps)
28     [bobmeasure] cs=4 & cb=bb → (cs'=5); // bob has right basis
29     [bobmeasure] cs=4 & cb≠bb → UNLUCKY : (cs'=5) & (cd'=1-cd) +
30     LUCKY : (cs'=5) & (cd'=cd); // wrong basis
31     [bobget] cs=5 → (cs'=0);
32 endmodule
33
34 module alice
35     as : [ 0..5 ]; // local state
36     ai : [ 1..N ]; // index
37     ab : [ 0..1 ]; // basis
38     ad : [ 0..1 ]; // data
39
40     // choose basis (randomly) (after eve has chosen her basis)
41     [ ] as=0 & es>0 → 0.5: (as'=1) & (ab'=0) + 0.5 : (as'=1) & (ab'=1);
42     // choose data (randomly)
```

```

43     []      as=1 → 0.5: ( as'=2 ) & ( ad'=0 ) + 0.5 : ( as'=2 ) & ( ad'=1 );
44     // send data to channel
45     [aliceput] as=2 → ( as'=3 );
46     // reveal bases/data to bob
47     [reveal] as=3 → ( as'=4 );
48     // loop ( go onto next bit , unless finished )
49     [loop] as=4 & ai<N → ( as'=0 ) & ( ai'=ai+ 1 ); // not finished
50     [loop] as=4 & ai=N → ( as'=5 ); // finished
51     // stop ( in either state 4 or 5 )
52     // ( depends whether bob comes to a conclusion before all the bits have been sent )
53     [stop] as=4 → ( as'=4 );
54     [stop] as=5 → ( as'=5 );
55 endmodule
56
57 module bob
58     bs : [ 0..7 ]; // local state
59     bb : [ 0..1 ]; // basis
60     bd : [ 0..1 ]; // data
61     bdn : bool init true ; // bob doesn't know
62
63     // choose basis ( randomly ) ( after eve has chosen her basis )
64     []      bs=0 & es>0 → 0.5 : ( bs'=1 ) & ( bb'=0 ) + 0.5 : ( bs'=1 ) & ( bb'=1 );
65     // read data from channel
66     [bobmeasure] bs=1 → ( bs'=2 );
67     [bobget] bs=2 → ( bs'=3 ) & ( bd'=cd );
68     // see alice 's bases/data
69     [reveal] bs=3 & bb≠ab → ( bs'=4 ); // bases different
70     [reveal] bs=3 & bb=ab & bd=ad → ( bs'=4 ) & ( bdn'=false ); // bases and data same
71     [reveal] bs=3 & bb=ab & bd≠ad → ( bs'=7 ); // bases same but data different
72     // loop ( go onto next bit , unless finished )
73     [loop] bs=4 & ai<N → ( bs'=0 ); // not finished
74     [loop] bs=4 & ai=N & bdn → ( bs'=5 ); // finished
75     [loop] bs=4 & ai=N & !bdn → ( bs'=6 ); // finished
76     // stop - don't know anything
77     [stop] bs=5 → ( bs'=5 );
78     // stop - nothing detected
79     [stop] bs=6 → ( bs'=6 );
80     // stop - detected eavesdropper
81     [stop] bs=7 → ( bs'=7 );
82 endmodule
83
84 module eve
85     es : [ 0..4 ]; // local state
86     eb : [ 0..1 ]; // basis
87     ed : [ 0..1 ]; // data
88
89     // choose basis
90     []      es=0 → ( es'=1 ) & ( eb'=0 );
91     []      es=0 → ( es'=1 ) & ( eb'=1 );
92     // read data from channel
93     [evemeasure] es=1 → ( es'=2 );
94     [eveget] es=2 → ( es'=3 ) & ( ed'=cd );

```

```
95         // send data to channel
96 [eveput] es=3 → (es'=0);
97 endmodule
```

---



## APPENDIX B

# Sample PRISM Output

Below is the output of PRISM for the single-bit version of BB84 with  $N = 64$ .

---

```
1 PRISM
2 =====
3 Version: 1.2
4 Loading modules file "qkd64new.nm"...
5 Building model...
6 MTBDD variables used (27r, 27c, 14nd): a0 a1 a2 a3 a4 a5 a6 a7 a8 channel.s
7 cs.0 cs'.0 cs.1 cs'.1 cs.2 cs'.2 cb.0 cb'.0 cd.0 cd'.0 alice.s as.0 as'.0
8 as.1 as'.1 as.2 as'.2 ai.0 ai'.0 ai.1 ai'.1 ai.2 ai'.2 ai.3 ai'.3 ai.4 ai'.4
9 ai.5 ai'.5 ab.0 ab'.0 ad.0 ad'.0 bob.s bs.0 bs'.0 bs.1 bs'.1 bs.2 bs'.2 bb.0
10 bb'.0 bd.0 bd'.0 bdn.0 bdn'.0 eve.s es.0 es'.0 es.1 es'.1 es.2 es'.2 eve.l4
11 eb.0 eb'.0 ed.0 ed'.0
12 Computing reachable states ...
13 Reachability : 770 iterations in 14.86 seconds (average 0.019299, setup 0.00)
14 Model built in 15.265 seconds.
15 Type:      Nondeterministic (MDP)
16 Modules:  channel alice bob eve
17 Variables : cs cb cd as ai ab ad bs bb bd bdn es eb ed
18 States :   210191
19 Transition matrix: 9744 nodes (3 terminal), 191789184 minterms
20 ODD: 1678 nodes
21 Loading properties file "qkd5new.pctl"...
22 6 PCTL formulas read:
23 (1)  $P \geq 0.5$  [ true U bs=5 ]
24 (2)  $P \geq 0.5$  [ true U bs=6 ]
25 (3)  $P \geq 0.5$  [ true U bs=7 ]
26 (4)  $P \leq 0.5$  [ true U bs=5 ]
27 (5)  $P \leq 0.5$  [ true U bs=6 ]
28 (6)  $P \leq 0.5$  [ true U bs=7 ]
29 Model checking property (1)...
30 (1)  $P \geq 0.5$  [ true U bs=5 ]
31 Engine: MTBDD
32 PCTL Until :
33 b1 = 210191 states , b2 = 48 states
34 yes = 48 states , no = 0 states , maybe = 210143 states
35 Computing remaining probabilities ...
36 Building iteration matrix MTBDD... [nodes=9690] [189.3 Kb]
37 Starting iterations ...
38 Iterative method: 588 iterations in 53.87 seconds (average 0.091582, setup 0.02)
39 Probabilities ( cs=0 & as=0 & ai=1 & bs=0 & es=0 ):
40 Number of states satisfying until : 1632
41 Model checking completed in 54.273 seconds.
```

42 Number of states satisfying PCTL **formula**: 1632 ( initial state not satisfied )  
43 Model checking property (2) ...  
44 (2)  $P \geq 0.5 [ \text{true} \ U \ bs=6 ]$   
45 Engine: MTBDD  
46 PCTL Until :  
47 b1 = 210191 states , b2 = 84 states  
48 yes = 84 states , no = 0 states , maybe = 210107 states  
49 Computing remaining probabilities ...  
50 Building iteration matrix MTBDD... [nodes=9620] [187.9 Kb]  
51 Starting iterations ...  
52 Iterative method: 770 iterations in 163.45 seconds (average 0.212234, setup 0.03)  
53 Probabilities ( cs=0 & as=0 & ai=1 & bs=0 & es=0):  
54 0: (0,0,0,0,1,0,0,0,0,1,0,0,0) = 1.9431905663704275E-4  
55 Number of states satisfying until : 12552  
56 Model checking completed in 163.921 seconds.  
57 Number of states satisfying PCTL **formula**: 12552 ( initial state not satisfied )  
58 Model checking property (3) ...  
59 (3)  $P \geq 0.5 [ \text{true} \ U \ bs=7 ]$   
60 Engine: MTBDD  
61 PCTL Until :  
62 b1 = 210191 states , b2 = 3048 states  
63 yes = 3048 states , no = 0 states , maybe = 207143 states  
64 Computing remaining probabilities ...  
65 Building iteration matrix MTBDD... [nodes=9524] [186.0 Kb]  
66 Starting iterations ...  
67 Iterative method: 769 iterations in 157.42 seconds (average 0.204668, setup 0.03)  
68 Probabilities ( cs=0 & as=0 & ai=1 & bs=0 & es=0):  
69 0: (0,0,0,0,1,0,0,0,0,1,0,0,0) = 0.9998056809433629  
70 Number of states satisfying until : 194815  
71 Model checking completed in 157.858 seconds.  
72 Number of states satisfying PCTL **formula**: 194815 (including initial state)  
73 Model checking property (4) ...  
74 (4)  $P \leq 0.5 [ \text{true} \ U \ bs=5 ]$   
75 Engine: MTBDD  
76 PCTL Until :  
77 b1 = 210191 states , b2 = 48 states  
78 yes = 48 states , no = 0 states , maybe = 210143 states  
79 Computing remaining probabilities ...  
80 Building iteration matrix MTBDD... [nodes=9690] [189.3 Kb]  
81 Starting iterations ...  
82 Iterative method: 587 iterations in 43.74 seconds (average 0.074463, setup 0.03)  
83 Probabilities ( cs=0 & as=0 & ai=1 & bs=0 & es=0):  
84 Number of states satisfying until : 209663  
85 Model checking completed in 44.09 seconds.  
86 Number of states satisfying PCTL **formula**: 209663 (including initial state)  
87 Model checking property (5) ...  
88 (5)  $P \leq 0.5 [ \text{true} \ U \ bs=6 ]$   
89 Engine: MTBDD  
90 PCTL Until :  
91 b1 = 210191 states , b2 = 84 states  
92 yes = 84 states , no = 0 states , maybe = 210107 states  
93 Computing remaining probabilities ...

94 Building iteration matrix MTBDD... [nodes=9620] [187.9 Kb]  
95 Starting iterations ...  
96 Iterative method: 769 iterations in 137.05 seconds (average 0.178192, setup 0.02)  
97 Probabilities ( cs=0 & as=0 & ai=1 & bs=0 & es=0):  
98 0: (0,0,0,0,1,0,0,0,0,0,1,0,0,0) =1.9431905663704275E-4  
99 Number of states satisfying until : 198223  
100 Model checking completed in 137.704 seconds.  
101 Number of states satisfying PCTL **formula**: 198223 (including initial state)  
102 Model checking property (6)...  
103 (6)  $P \leq 0.5$  [ true U bs=7 ]  
104 Engine: MTBDD  
105 PCTL Until :  
106 b1 = 210191 states , b2 = 3048 states  
107 yes = 3048 states , no = 0 states , maybe = 207143 states  
108 Computing remaining probabilities ...  
109 Building iteration matrix MTBDD... [nodes=9524] [186.0 Kb]  
110 Starting iterations ...  
111 Iterative method: 768 iterations in 124.71 seconds (average 0.162357, setup 0.02)  
112 Probabilities ( cs=0 & as=0 & ai=1 & bs=0 & es=0):  
113 0: (0,0,0,0,1,0,0,0,0,0,1,0,0,0) =0.9998056809433629  
114 Number of states satisfying until : 15720  
115 Model checking completed in 125.135 seconds.  
116 Number of states satisfying PCTL **formula**: 15720 ( initial state not satisfied )

---



## APPENDIX C

# Shell Scripts for Processing PRISM Output

### constr.sh

---

```
1 # Input arguments to this script are : number of bits and number of property to be checked
2 NUM=$1
3 # Construct model file for $NUM bits:
4 echo "nondeterministic" >qkd$NUM.nm
5 echo "const N = $NUM;" >>qkd$NUM.nm
6 echo "" >>qkd$NUM.nm
7 cat qkdbare.nm >>qkd$NUM.nm
8
9 # Run PRISM to check property $P of the model:
10 P=$2
11 echo "Checking property $P of qkd$NUM.nm ..."
12 echo "Results will be placed in file results-prop$P-$NUM-bits.txt."
13 prism -nopre -nofair -m -prop $P qkd$NUM.nm qkd5new.pctl >results-prop$P-$NUM-bits.txt
```

---

### modelcheck1to64.sh

---

```
1 #!/bin/bash
2 # Script to run PRISM on models from 5 bits to 64 bits .
3 # Nikolaos Papanikolaou
4 BITS=5;
5 while [ "$BITS" -le 64 ]
6 do
7     echo "CHECKING PROPERTY 1 (NOT DETECTING ANYTHING) FOR $BITS BITS"
8     ./constr.sh $BITS 1
9     let BITS="$BITS"+1;
10 done
11 while [ "$BITS" -le 64 ]
12 do
13     echo "CHECKING PROPERTY 2 (NOT DETECTING EAVESDROPPING) FOR $BITS BITS"
14     ./constr.sh $BITS 2
15     let BITS="$BITS"+1;
16 done
17 while [ "$BITS" -le 64 ]
18 do
19     echo "CHECKING PROPERTY 3 (DETECTING EAVESDROPPING) FOR $BITS BITS"
20     ./constr.sh $BITS 3
21     let BITS="$BITS"+1;
22 done
```

---

### retrieveprobs.sh

```

1 #!/bin/bash
2 # Script to extract probabilities from PRISM output
3 # Nikolaos Papanikolaou
4 PROP=1;
5 while [ "$PROP" -le 3 ]
6 do
7     touch "property$PROP-probs.txt"
8     BITS=5;
9     while [ "$BITS" -le 64 ]
10    do
11        FILENAME="results-prop$PROP-$BITS-bits.txt ";
12        PROBABILITY=$(cat $FILENAME | grep 0: | cut -d= -f2)
13        echo "$BITS $PROBABILITY" >>property$PROP-probs.txt
14        let BITS="$BITS"+1;
15    done
16 let PROP="$PROP"+1;
17 done

```

---

### retrievetimes.sh

```

1 #!/bin/bash
2 # Script to extract timings from PRISM output
3 # Nikolaos Papanikolaou
4 PROP=1;
5 while [ "$PROP" -le 3 ]
6 do
7     touch "property$PROP-probs.txt"
8     BITS=5;
9     while [ "$BITS" -le 64 ]
10    do
11        FILENAME="results-prop$PROP-$BITS-bits.txt ";
12        TIME=$(cat $FILENAME | grep 'Model checking completed' | cut -d ' ' - f5)
13        echo "$BITS $TIME" >>property$PROP-times.txt
14        let BITS="$BITS"+1;
15    done
16 let PROP="$PROP"+1;
17 done

```

---