The University of Warwick

TECHNIQUES FOR DESIGN AND VALIDATION OF QUANTUM PROTOCOLS

Nikolaos K. Papanikolaou

A THESIS SUBMITTED TO
THE DEPARTMENT OF COMPUTER SCIENCE
IN CANDIDACY FOR THE DEGREE OF
MASTER OF SCIENCE BY RESEARCH

Academic Supervisor: *Rajagopal Nagarajan*

Coventry, September 2004

Typeface: URW Palatino 11/13
Typesetting: LaTeX 2ε
The typography of this document is dedicated to Jan Tschichold (1902-1974), *in memoriam.*

To my parents

Contents

List of Illustrations

List of Tables

Preface

No greater need has ever arisen in human societies than the need for effective communication. In the information age that we live in, more than ever in history, we are faced with the problem of exchanging data quickly and accurately. Enabling technologies, such as global inter–networks, are partly a remedy for this problem and partly a cause for its aggravation; for the high speed of such tools as electronic mail and on–line video conferencing helps to raise people's expectations.

At the same time, the educated layman is becoming increasingly aware of the need for *secure* communication. Purchasing items from the Internet is almost an everyday task for many people, yet the security of such a transaction is commonly regarded as dubious. In recognition of the security threats associated with computer technology, businesses are investing greater and greater sums to ensure the integrity and privacy of corporate data. To quote from a recent Microsoft Progress Report[1], by Bill Gates:

> " Security is as big and important a challenge as any our industry has ever tackled. It is not a case of simply fixing a few vulnerabilities and moving on. Reducing the impact of viruses and worms to an acceptable level requires fundamentally new thinking about software quality, continuous improvement in tools and processes, and ongoing investments in resilient new security technologies designed to block malicious or destructive software code before it can wreak havoc. It also requires computer users to be proactive about deploying and managing products. "

Actually, the task of transferring information from one place to another integrally, efficiently and securely is as old as mankind itself. A rudimentary instance of this task is fire signalling, used by ancient civilisations such as the Greeks to communicate the occurrence of a particular event. Consider, for a moment, the use of a fire signal to announce victory in a battle. A fire signal expresses an *integral* message, in the sense that it has a clear, unambiguous meaning, namely that a battle has been won. However, it is *inefficient*, since it is not certain that the message will be received (a receiver, located in a different city, would have to be constantly looking out for a fire in a specific place), and there is trouble in setting it up. Finally, a fire signal is an *insecure* kind of communication, in the sense that the enemy can intercept it inasmuch as the intended receiver.

It is widely believed that enforcing a discipline on communication is an effective means of satisfying the requirements of integrity, efficiency and security. Such a discipline is known as a *protocol*. Protocol is a concept that originates in international diplomacy, where it is used to establish a commonly agreed set of rules between representatives of different countries and cultures; these rules enable fruitful cooperation.

The term *protocol* was introduced into computer science discourse in 1967 by R.A. Scantlebury and K.A. Bartlett at the National Physical Laboratory in England. They used it in a memorandum entitled "A protocol for use in the NPL data communications network"[2]. In this context, a protocol is a procedure which enforces a discipline on the form and sequencing of data, as well as the exact meaning of possible messages.

Since then, data networks have taken many forms and sizes, and protocols have become progressively more complex. This complexity has made protocols extremely error–prone. In this respect, protocols may be likened to computer algorithms. Algorithms also specify a *disciplined* procedure for solving a problem; moreover, they have to be defined in such a way as to handle unexpected scenarios, for example erroneous input.

The obvious way to test how prone a given algorithm is to error is to run it on all possible inputs and observe its output. There are at least two major problems with this approach:

1. One has to imagine all the possible inputs that could be given to the algorithm.

2. It is enormously time–consuming, if not impossible, to exhaustively test an algorithm.

These problems can only really be addressed for small, simple algorithms. Moreover, the same problems arise in the development of protocols.

Therefore, methods for rigorous definition and testing of protocols and algorithms are essential. In the last ten years, the field of *formal methods* has flourished in response to precisely that need. This has led to the development of numerous specification languages and automated testing tools. It was precisely such a tool, FDR, that helped Gavin Lowe discover a flaw in the widely used Needham–Schroeder Public–Key authentication protocol[3]. This is heralded as one of the greatest successes of the formal approach to system design and validation.

The application of formal methods to protocols for computer networks was studied extensively by Gerard Holzmann in the 1980s and culminated in the creation of an automated verification tool, SPIN[4]. In April 2002, the Association for Computing Machinery awarded to the creator of SPIN the System Software Award for 2001. This is a very strong indication of the importance and utility of SPIN and similar validation tools.

In September, 2003, Holzmann was interviewed for Fawcette Technical Publications on the subject of SPIN[5]. His words on the utility of this tool for practical applications are distinctive:

> " At Lucent Technologies we used SPIN over a period of two years to check that the call–processing software for a new telephone switch was free of concurrency–related bugs. The code was heavily multi–threaded, as most applications are these days. The call–processing code was also considered to be the most difficult part of the switch to get right: it really forms the heart of the switch. [...]
>
> During the lifetime of this product, not a single software defect was reported in this part of the code: a highly unusual phenomenon, and a testament to the effectiveness of the SPIN–based

> approach to testing. Today, SPIN tends to be used mostly for critical software development, and more sporadically in routine software development. At NASA/JPL, for instance, SPIN is used to check that the critical components of flight software for space missions are bug–free. "

While computer scientists and engineers have been witnessing the enormous growth of their field, from the age of room–sized computers to the age of "model checking" with SPIN another revolution has been taking place in physics. The world of the atom is no longer a theoretical curiosity, but is getting more and more amenable to direct observation. Through the synergy of physicists and computer scientists, we have come to realise that quantum–mechanical effects can be put to use for efficient computation, as well as for secure communication. It is this last discovery of *quantum cryptography* that will concern us mostly in this thesis, whose objective is to demonstrate the application of formal methods to protocols for quantum communication.

Quantum Technology

One of the greatest visionaries that ever lived was the Nobel Prize-winning physicist Richard P. Feynman. His renowned talk, "There's Plenty of Room at the Bottom"[6], spawned a revolution in physical science. Feynman's ideas are widely regarded as the precursor to nanotechnology, since he suggested that there was no fundamental obstacle to "writing the entire 24 volumes of the Encyclopaedia Britannica on the head of a pin". More than 40 years later, we are witnessing an endless miniaturisation of electronic circuits, an ever–increasing mass of information to handle, and a great variety of technologies for its storage. Feynman's suggestion is definitely not nearly as absurd today as it was in 1959, when it was made. A naïve extrapolation of Moore's Law leads us to conclude that we will be able to transcribe a single bit of information on an atom in about 20 years from now[7]. This is no trivial issue for, the ability to manipulate objects on the atomic scale while coping with the associated quantum mechanical phenomena is a promise physicists have yet to fulfil. In order to store data onto individual particles, highly advanced experimental methods are needed.

It follows from the above that the transition to quantum technology is an inevitable consequence of miniaturisation. In addition to this, quantum–mechanical phenomena may be exploited so as to perform feats of computation hitherto considered impossible. Therefore, an understanding of such phenomena is of great importance to the computer scientist.

The very notion that fundamental elements of matter could be used for computational purposes has led to the study of *quantum computers*. If ever such devices were constructed, *quantum algorithms* to perform prime factorisation and inverse database searching efficiently could be implemented. The ability to compute the prime factors of a given number is of great significance, since this would permit the inversion and direct violation of several cryptographic systems in common use today. The interested reader is referred to the standard literature on the subject for more details[8].

A clear distinction is made by researchers in the field between the study of quantum *computation* and the study of quantum *information*. The latter is concerned with such matters as the capacity of quantum channels (i.e. communication channels operating on quantum–mechanical principles) and rates of transmission on these channels. Furthermore, the theory of quantum information also deals with quantum cryptography and various proofs of its security. Schemes for

exchanging information using quantum–mechanical phenomena (e.g. entanglement) are referred to as modes of *quantum communication*.

If a new name should be given to the era we are entering, it definitely has to be "The Era of Quantum Technology". The combined advancement of quantum computation and quantum information theory will be its main novelty.

Implications of Quantum Technology

In order to come to terms with the challenge posed by this advancement, an in–depth analysis of fundamental physical phenomena is necessary. Such an analysis undoubtedly will lead to a better understanding, and potentially the resolution, of many open problems in quantum theory. Several conundrums have arisen from the theory, and even Albert Einstein was never completely satisfied with its predictions. For instance, at the atomic level, nature exhibits randomness; we can never know with certainty the exact position of an electron orbiting around an atomic nucleus. Quantum theory provides rules for obtaining the *probability* of finding the electron at a particular point in space. The widespread belief that the universe is causal, which had been upheld since the time of Newton, has collapsed in the face of this apparent randomness. Einstein is often remembered for having said "I do not believe that God plays dice"[9]. This is a philosophical claim that cannot be proved, but if we are to accept the quantum theory, we have to regard it as false. Developments in the area of quantum computation and information are likely to provide insights into such foundational issues.

It goes without saying that these developments will create a closer bond between the disciplines of physical and computer science. This bond is far from artificial, since it has long been recognised that the centerpiece of computer science, digital information, is inherently a manifestation of nature; this is expressed clearly in Rolf Landauer's dictum "Information is inevitably physical"[10].

The Role of Computer Science

The implementation of quantum computers and quantum information carriers clearly impinges on experimental physics and the ability to harness the effects that occur at the atomic level. It is therefore largely the duty of physicists to provide a foundation for such developments. This does not diminish, however, the role of computer scientists in the growth of this new field. The discipline of computing has much to offer in terms of design techniques and mathematical principles.

Since quantum–mechanical devices are still mostly a matter of theoretical speculation, we can only imagine or attempt to simulate their behaviour. Existing methods for system modelling and analysis are useful and applicable to this endeavour. Of particular importance to the subject matter of this work is the practice of formal methods. As mentioned at the beginning of this chapter, these include a number of notations and software tools that can be used for designing and testing abstract models.

Aims and Objectives of This Work

The present work is concerned with the detailed study of cryptographic and general communication protocols that involve the exploitation of quantum–mechanical effects. In particular, all such protocols will be identified and recognised as a class. Furthermore, these *quantum protocols* will be modelled and analysed carefully using existing software tools. The spin software mentioned in previous sections will be considered to this end, as well as the prism tool developed at the University of Birmingham. The results of analyses made using these tools will be presented. Attention will be paid to the limitations of existing tools and languages in view of the particularities of quantum protocols.

The focus then turns to the development of a new programming language and associated software interpreters, suited especially for the formal description and the simulation of quantum protocols. Connections will be made with existing "quantum programming languages", and potential extensions of the core language (e.g. to account for future developments) will be considered.

Acknowledgements

I am greatly indebted to my supervisor, Dr. Rajagopal Nagarajan, for his friendly encouragement and support for the duration of this work. Also, I wish to express my gratitude to Dr. Jane Sinclair, who consented to review this document prior to submission. I also thank Dr. Huibiao Zhu for useful discussions on the concept of process bisimulation.

A great many thanks are due to Dr. David Parker and Dr. Gethin Norman, currently with the Department of Computer Science at the University of Birmingham, who provided much needed assistance with the prism tool.

Abstract

The objective of this thesis is to perform formal specification and automated validation of the class of protocols associated with quantum cryptography.

Current analyses and proofs regarding quantum cryptographic protocols and their security are information-theoretic and also require in–depth understanding of the underlying physics. The alternative approach presented here involves the use of computer modelling languages and verification software. In particular, the *logical* model checker SPIN[11] and the *probabilistic* model checker PRISM[12] are used to analyse quantum key distribution, entanglement and quantum teleportation.

The value of our approach lies not only in the fact that it is conceptually simpler than existing proofs, but in that it allows us to disambiguate protocol definitions and to assess their properties in various circumstances. By varying the values of certain parameters, different attacks on the protocols can be attempted and implementation–specific attributes can be analysed.

The quantum key distribution protocol[13] BB84 has been proved to be unconditionally secure against all types of eavesdropping[14]. Although a general proof of this result is extremely hard to generate automatically, it is possible to develop specific protocol attacks and validate the protocol by computer. Assuming two types of eavesdropping attack (intercept–resend and random–substitute), we have used the PRISM tool to compute exactly the probability of successful eavesdropping, and found it to diminish as the number of transmitted qubits is increased. This result is linked to Mayers' security criteria[15] for BB84. Also, we briefly describe a new quantum protocol definition language, named QSPEC[16]. The recent CQP process algebra, due to Nagarajan and Gay[17], is reviewed and examples of its application are provided. Alternative formalisms and validation tools are discussed, including *QPAlg*, PROBMELA and *probUSM*[18], the logic of knowledge and time for quantum systems due to Van der Meyden[19], and the QCSim simulator[20].

1

Introduction and Preview

In the 1980s it was realised for the first time that quantum-mechanical phenomena may be used directly for manipulating, storing and communicating digital information. Of major importance was the discovery of quantum algorithms for prime factorisation and inverse search, which outperformed the best known classical algorithms significantly[1]. However, to implement any one of these algorithms, a quantum-mechanical computer is necessary. To date, little progress has been made in constructing a full–scale quantum computer.

The storage and manipulation of digital information form the centerpiece of computer science. Theoretical computer scientists tend to regard information as an abstract entity, with an existence above and beyond hardware and software. It was not long ago, however, that this view was refuted with Landauer's observation that information is inevitably tied to a physical representation[2]. Landauer argued that information could only exist when manifested in some physical form, and this was good cause for a paradigm shift; there was evidence of an important link between physics and computing.

Programmers and computer users alike are familiar with *Moore's law,* which is concerned with the rapid increase in the power of computer hardware over time. The trend it predicts has been upheld by computer hardware since Moore's time and, it is believed, is likely to continue to do so for the foreseeable future. If this does occur, the continuous miniaturisation of electronic circuits will soon lead to the subatomic scale. There will come a time when a single bit of information will be transcribed onto an object the size of an atom. In order to manipulate such minute objects, knowledge and application of quantum theory will be necessary. The need will also arise for effective physical means of storing, transforming and transmitting these objects.

Clearly, it is valuable to investigate the possibilities that quantum theory has to offer for computational purposes. Using various aspects of the quantum world, data can be represented in a very effective manner, and computations can be made massively parallel. Testament to this observation are the quantum algorithms discovered for factorisation and inverse database search, which both provide major improvements over methods currently in use[3].

Research in the field of quantum *information* has led to the observation that quantum-mechanical effects can provide not only increased computational efficiency, but can also be used to expedite the transmission of information in a highly secure manner.

Stephen Wiesner[4] conceived of communication channels with no analogue in classical physics and computing using such effects. This paved the way for quantum *cryptography*, allowing transmissions which it is impossible, in principle, to eavesdrop. Since the observation of any quantum system is destructive, any attempt to obtain information encoded therein can be detected. This is in contrast to conventional transmission media, on which it is possible to eavesdrop *passively*, thus going completely unnoticed.

It has been shown that the standard protocol for quantum key distribution, "BB84," is secure against all possible attacks[5]. Therefore, with respect to our current understanding of quantum theory, BB84 is immune against eavesdropping and there is no known way to systematically subvert a perfect implementation of it[6].

Since the discovery of the efficient prime factorisation algorithm, it has been acknowledged that a quantum computer would likely be capable of easily breaking cryptosystems in use today, such as RSA. The possibility of quantum cryptography is a relief in this sense, because it signifies the emergence of a new kind of cryptography, which cannot be broken even using a quantum computer.

Whereas quantum computers are still mostly a theoretical possibility, the techniques of quantum cryptography have been implemented experimentally and shown to work as expected. Quantum key distribution has been performed over optical fibres as well as over open air. A great amount of research has been initiated to create practical quantum cryptography systems[7].

The idea of a quantum *network* has arisen as a valuable possibility for defence as well as for commercial purposes. Since perfectly secure quantum communication links can be made, data transfer will be improved greatly. The DARPA agency in the United States is investigating such a possibility, and a European research consortium ("SECOQC") has been established for similar purposes.

1.1 The Wire-Tap Channel

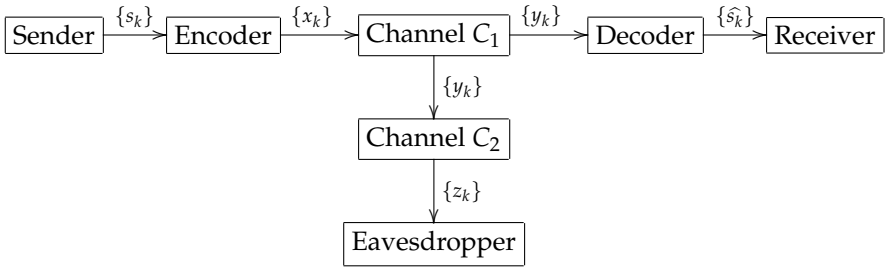
Quantum cryptography is an attempt to provide a solution to the problem of secure data transmission. Suppose that a data sequence is to be transmitted over a communications channel. At the receiver, there is a wire–tapper or *eavesdropper* which is trying to gain unauthorised access to the data. The issue that needs to be tackled is whether there exists a way of preventing the eavesdropper from succeeding, while transmitting the data reliably to the receiver.

In order to prevent corruption of the data due to imperfections in the transmission medium, some form of *channel coding* is necessary. Also, a cryptographic mechanism must be put in place so as to avert the eavesdropper. As we shall see shortly, it is possible to devise a coding scheme that fulfils the objectives of cryptography as well. Before discussing this, it is valuable to phrase the problem of secure transmission in more formal terms.

The situation can be described as follows: we have a noisy communication system that is being wire–tapped through a second noisy channel. The transmitter, or *source*, emits a sequence of bits $\{s_k\}$. The sequence is encoded, using a suitable scheme, into a binary sequence $\{x_k\}$. The communication channel C_1 induces a certain number of errors, and the channel output is, thus, yet a new sequence $\{y_k\}$. At the receiving end, the sequence $\{y_k\}$ is subjected to a wire–tap through a second noisy channel C_2, supplying the eavesdropper with bits $\{z_k\}$. Finally, the legit-

imate receiver decodes the output of C_1 to recover, at least partly, the transmitted sequence. As a consequence of channel errors the decoded sequence $\{\widehat{s_k}\}$ differs from the transmitted sequence in certain positions. The situation is shown diagrammatically in Figure 1.1.

Figure 1.1 Wyner's wire–tap channel.



We define the probability of error as the quantity

$$P_e = \frac{1}{N} \sum_{k=1}^{N} \Pr\{\widehat{s_k} \neq s_k\} \tag{1.1}$$

where N is the length of the transmitted sequence. Clearly, a system designer would be concerned with developing a coding scheme that minimises P_e. For this purpose there exists a great number of error–correcting codes.

The most common model for a noisy channel that carries binary data is the *binary symmetric channel*, which is characterised by a *crossover probability* p_0. This quantity determines the chance that a '0' is flipped into a '1' by the channel and vice versa. The behaviour of a binary symmetric channel with crossover probability p_0 is described by the following expression, which relates the output of the channel to its input[8]:

$$\Pr\{Y_n = y \,|\, X_n = x\} = (1 - p_0) \cdot \delta_{xy} + p_0 \cdot (1 - \delta_{xy}) \tag{1.2}$$

If we assume that the noisy channels in the system of Figure 1.1 are binary symmetric, it is easy to deduce that the eavesdropper is likely to encounter difficulty in obtaining the transmitted sequence. The greater the crossover probability of the two channels, the greater the chance of receiving a corrupt bit value and thence of errors in the intercepted sequence $\{z_k\}$. Even when an error correction scheme is employed, the eavesdropper does not have access to the decoded sequence $\{\widehat{s_k}\}$ and can only conjecture as to the content of the transmitted message.

Another objective of the system designer will therefore be to maximize the level of uncertainty and confusion of the eavesdropper, which corresponds to the so–called *equivocation of the data as seen by the eavesdropper.* The equivocation Δ quantifies the uncertainty of the eavesdropper about the sequence $\{y_k\}$ given the result $\{z_k\}$ of her observations. Its value is, in general,

$$\Delta = \frac{1}{N} \cdot H(y_1, \ldots, y_N \,|\, z_1, \ldots, z_N) \tag{1.3}$$

where H is the Shannon conditional entropy associated with the wire–tap channel C_2, and N is

the length of the transmitted sequence.

The above presentation realistically assumes the presence of noise in the communication system. Noise causes difficulties not only to the eavesdropper, but also to the legitimate receiver of the message. Noise can be effectively combatted using error–correcting codes. The problem that remains is how to code messages to ensure secrecy. In the case of a perfect (i.e. noiseless) channel, it is possible for an eavesdropper to intercept an entire transmission successfully.

Wyner (1975) discusses in detail the situation presented here and shows that there exists a tradeoff between the rate of transmission and the equivocation of the eavesdropper. In order to maximise an eavesdropper's uncertainty, a coding scheme necessarily reduces the transmission rate. An excellent exposition of the general coding problem and the theoretical foundations of cryptography is that of Welsh (1998). Also, Hamming (1986) discusses classical error–correcting codes and information theory in general.

1.2 CONJUGATE CODING

Suppose that there was a way of coding data that:

- prevents, with arbitrarily high probability, an eavesdropper from intercepting transmissions correctly;
- makes manifest to the legitimate users of the communication channel the presence of an eavesdropper.

Such a coding scheme would be capable of resolving the problem presented in the previous section.

A scheme of this kind was suggested by S. Wiesner and is known as *conjugate coding*[9]. The principle is to represent the data to be transmitted using the states of elementary quantum mechanical systems. For instance, each bit in a binary sequence can be represented using a hydrogen atom, with a '0' corresponding to the atom's ground state and a '1' to its excited state. Other examples of two–level quantum systems that can be used to represent binary values are:

- Spin–$\frac{1}{2}$ particles (*spin* is an attribute of particles, for our purposes a kind of angular momentum with two discrete values, $+\frac{\hbar}{2}$ and $-\frac{\hbar}{2}$);
- Polarised photons (discussed at length in section 1.2.2).

Thus, information is conveyed by variations in a quantum state — and quantum states exhibit several intriguing properties.

1.2.1 FUNDAMENTAL PROPERTIES OF QUANTUM SYSTEMS

A two–level quantum system such as a hydrogen atom or spin–$\frac{1}{2}$ particle can exist in a state which is a *superposition* of its two *basis states.* If the basis states are taken to represent bit values of '0' and '1', the superpositions correspond to "mixtures" of '0' and '1'; it is possible for a quantum system to denote a '0' and a '1' *simultaneously*. This observation leads to the possibility of massive

computational parallelism, which is where the emphasis is put in quantum *computation*. For our purposes it suffices to note that the state of an elementary two–level quantum system can be written as a linear combination of its two basis states:

$$|\psi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle \tag{1.4}$$

A quantum state is conventionally written as a vector in the Dirac notation, i.e. in the form $|\psi\rangle$. For a system with two levels, $|\psi\rangle$ belongs to a two–dimensional complex vector space $\mathscr{H}_2$, which is spanned by basis states $|0\rangle$ and $|1\rangle$. Also note that α and β are complex numbers with $|\alpha|^2 + |\beta|^2 = 1$. A two–level quantum system is referred to as a *quantum bit,* or *qubit.*

Another crucial aspect of quantum states is that their value at any given moment cannot be obtained directly. In fact, it is not possible to observe a quantum system without irrevocably perturbing its state. The theory predicts that a system described by (**1.4**) will, when measured with respect to $|0\rangle$ and $|1\rangle$, change its state to one of $|0\rangle$ or $|1\rangle$ at random. This randomness is inherent in Nature; the coefficients α and β above are the square roots of the probabilities of obtaining $|0\rangle$ or $|1\rangle$, respectively. We refer to this phenomenon as *probabilistic measurement*.

There is a further complication associated with measurement. Measurement is always performed *with respect to* a set of basis states. A two–dimensional vector space can be spanned by arbitrary pairs of vectors, not only $|0\rangle$ and $|1\rangle$ above. A different basis, say $\{|+\rangle, |-\rangle\}$, could be used instead, as long as the basis vectors $|+\rangle$ and $|-\rangle$ are mutually orthogonal. It is just as plausible to measure a state of the form (**1.4**) with respect to the basis $\{|+\rangle, |-\rangle\}$. In this case, quantum theory predicts that the result of the measurement will be one of $|+\rangle$ or $|-\rangle$, at random. We use the term *conjugate* to describe bases with the property defined below:

Definition 1.1 (Brassard (1988)) *Two bases are termed* conjugate *if a system prepared in a specific state of one basis will behave entirely randomly, and lose all its stored information, when subjected to measurement corresponding to the other basis.*

It is possible to alter states without performing a measurement. On one hand, quantum systems evolve naturally in accordance with the so–called *Schrödinger equation;* on the other hand, *unitary transformations* may be applied to states in order to manipulate them directly. In direct analogy to Boolean gates, quantum algorithms and protocols make use of *quantum gates*, which are predefined unitary transformations.

In a classic paper for physicists[10], it was proved that quantum states are impossible to clone. Through an intuitive argument it is easy to understand why this is true. In order to replicate the state of a given quantum system, that state has to be known in the first place. However, only a measurement can yield useful information about a quantum system and measurement changes the current state. So it follows that *it is impossible to copy a quantum state.*

Let it be noted that transformations on quantum states have a property with no classical counterpart: they are *reversible;* thus, any transformation can be undone. A Boolean NOT gate *is* reversible, because it is always possible to compute the value of the input for a given output; but the same cannot be done for the AND gate, whose single output is not enough to determine what the inputs were. Reversibility is a property which is ensured by representing all quantum operations by unitary[11] matrices.

1.2.2 Oblivious Transfer and Quantum Money

Wiesner (1969) describes two specific applications of conjugate coding. The first is "a means for transmitting two messages either but not both of which may be received", and is termed *quantum oblivious transfer.* The other is a *gedanken* experiment, in which the use of conjugate coding allows for the creation of banknotes which cannot be duplicated. Both applications use photons as information carriers; the polarisation of the photons varies according to the data being transmitted. A review of the nature of polarised photons is therefore in order.

Polarised Photons

Photons are examples of elementary quantum systems; any beam of light consists of a multitude of photons. In order to represent bits using photons, their *polarisation* is varied. For any given electromagnetic wave, the polarisation is the angle, with respect to the horizontal, of the plane in which that wave oscillates. A photon is perceived as an electromagnetic wave, and its polarisation can be fixed by passing it through special apparatus, such as a Polaroid filter or a calcite crystal.

A polarising apparatus has an *orientation;* this determines the axis of polarisation of the light beam emanating from it. It is even possible to isolate individual photons from the light beam.

According to the uncertainty principle of quantum mechanics (see Chapter 2), measurements on any single photon cannot reveal more than one bit of information regarding its polarisation. Consider a light beam polarised with axis ϑ. If this beam is sent into a filter oriented at a different angle φ, the photons will behave in a dichotomous and probabilistic fashion. With probability $\cos^2(\vartheta - \varphi)$, the filter will allow a photon to pass through it, or absorb it with probability $\sin^2(\vartheta - \varphi)$. Passing the photon through the filter demonstrates the destructive nature of measurement; all photons that emerge from the filter are affected by it and are all polarised at angle φ. The physics of the situation are explained in Figure 1.2.

To make this absolutely clear, we quote Bennett and Brassard (1984):

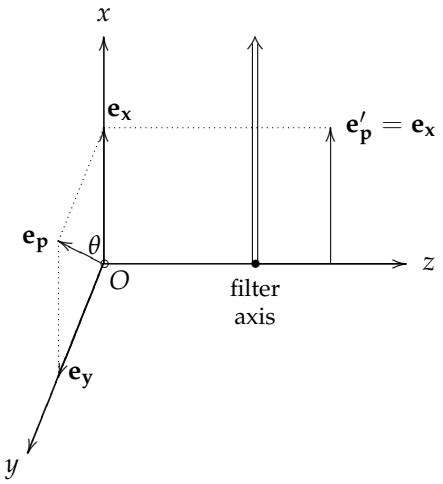
> " ... photons behave deterministically only when the two axes [of the photons and the polarising filter] are parallel (certain transmission) or perpendicular (certain absorption). If the two axes are not perpendicular, so that some photons are transmitted, one might hope to learn additional information about α [corresponding to ϑ above] by measuring the transmitted photons again with a polariser oriented at some third angle; but this is to no avail, because the transmitted photons, in passing through the β [corresponding to φ above] polariser, emerge with exactly β polarisation, having lost all memory of their previous polarisation α. "

As discussed above for the case of any two–level quantum system, the state space for a single polarised photon has two dimensions. Its state can be completely described as a linear combination of two unit vectors, e.g. $|0\rangle = (1,0)$ and $|1\rangle = (0,1)$. Take a photon polarised at angle ϑ to the horizontal; its state is described by the vector $(\cos\vartheta, \sin\vartheta)$. A measurement of this photon will yield state $|0\rangle$ with probability $\cos^2\vartheta$ or state $|1\rangle$ with probability $\sin^2\vartheta$.

Figure 1.2 An illustration of the effect of a polarising filter. A beam of light with polarisation $\mathbf{e_p}$ propagates along the direction Oz. The electric field for the beam is given by $\mathbf{E}(\mathbf{r}, t) = E_0 \cdot \mathbf{e_p} \cdot e^{i(kz-\omega t)}$, where E_0 is a constant. After the beam passes through the polariser, whose axis is parallel to Ox, only the x-component of polarisation remains. The electric field becomes

$$\mathbf{E}'(\mathbf{r}, t) = E_0' \cdot \mathbf{e_p'} \cdot e^{i(kz-\omega t)} = E_0' \cdot \mathbf{e_x} \cdot e^{i(kz-\omega t)}$$

Note that $\mathbf{e_p} = \mathbf{e_x} \cos\theta + \mathbf{e_y} \sin\theta$.



An alternative set of basis vectors for the polarised photon's state space is

$$|+\rangle = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) \tag{1.5a}$$

$$|-\rangle = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right) \tag{1.5b}$$

in which $|+\rangle$ corresponds to a 45° photon, and $|-\rangle$ to a 135° photon. By convention, the set of vectors $\{|0\rangle, |1\rangle\}$ is referred to as the *rectilinear basis*, while $\{|+\rangle, |-\rangle\}$ is referred to as the *diagonal basis:*

Notation 1.1 *The rectilinear basis, consisting of the vectors $|0\rangle$ and $|1\rangle$, is denoted by $\boxplus$.*

Notation 1.2 *The diagonal basis, consisting of the vectors $|+\rangle$ and $|-\rangle$, is denoted by $\boxtimes$.*

Notation 1.3 *A qubit is expressed here either using the rectilinear basis, as in $\alpha|0\rangle + \beta|1\rangle$, or using the diagonal basis, as in $\alpha|+\rangle + \beta|-\rangle$.*

Proposition 1.1 *The rectilinear and diagonal bases are conjugate.*

Proof. *The vectors in the two bases are mutually orthogonal; hence, measuring a state in the rectilinear basis with respect to the diagonal basis will yield an uncertain result (and vice versa); this is the distinctive property of conjugate bases.* ■

"Quantum Multiplexing", or Quantum Oblivious Transfer

Oblivious transfer, a term due to Rabin[12], is a procedure that makes it possible to multiplex two messages in a way that allows only one of them to be retrieved. Wiesner formulated such a procedure ("quantum multiplexing") long before Rabin coined the term for it, and he used the properties of polarised photons.

In this scheme, the transmitter emits a single, polarised burst of light onto an optical fibre for each bit in the two messages. Before emitting each burst, one of the two messages is chosen at random. If the first message is chosen, the i-th burst is fixed in the rectilinear polarisation basis. If the second message is chosen, the i-th burst is fixed in the diagonal polarisation basis.

The receiver uses a filter to measure the photons received. To obtain the first message, the filter is oriented horizontally or vertically (rectilinear case). To receive the second message, the orientation is shifted by 45° (diagonal case).

It is clear that, if the orientation of the receiving filter does not match the polarisation axis of a particular photon, the result of the measurement will be random. Therefore only one of the two transmitted messages can be correctly received with 100% accuracy.

"Quantum Money"

Using the principle of conjugate coding, it is possible to envisage banknotes which are impossible to counterfeit. It must be noted, however, that "quantum money" is not currently practicable, and thus has to be regarded a *gedanken* experiment.

Upon a quantum banknote are affixed several two–level systems. Rather than polarised photons, spin–$\frac{1}{2}$ particles are used. A spin of $\frac{\hbar}{2}$ represents '1', and a spin of $-\frac{\hbar}{2}$ represents '0'.

If a counterfeiter tries, for instance, to measure the particles on the banknote with basis $\boxplus$, although some measurements will yield a valid result, those particles encoded using $\boxtimes$ will be perturbed and the result will be random. Not only will the counterfeiter produce an invalid banknote, but the very attempt will eventually be detected at the mint, which is the only place the original bases used are known.

1.3 Quantum Key Distribution

We have seen that the use of quantum–mechanical phenomena enables curious means of data transmission without rival in classical communication. What remains to be shown is how conjugate coding can be applied in the context of the wire–tap channel. We will see how conjugate coding can prevent an eavesdropper, with arbitrarily high probability, from successfully intercepting a transmission.

To perform conjugate coding, the sender and receiver of a message must be linked through a quantum channel, that is, a channel designed to carry intact quantum particles, such as photons. Consider an altered version of the wire–tap system of Figure 1.1, with C_1 and C_2 replaced by quantum channels. Such a system will allow the sender to encode messages using quantum particles, while the eavesdropper will be capable only of destructively intercepting particles and transmitting substitutes to the receiver. The eavesdropper will not be able to re–transmit, or replicate the particles received, since unknown quantum states cannot be copied.

Suppose the sender wishes to transmit the sequence '101'. Each bit in this message is coded using one of the conjugate bases ⊞ or ⊠ chosen at random. In order to correctly decode the received particles, the basis used for encoding will be needed for measurement. Where incorrect bases are chosen, the result will be random.

Example 1.1 *Assume that the state $|\chi\rangle = |0\rangle$ is transmitted:*

- *If the receiver chooses to decode $|\chi\rangle$ using the ⊞ basis, the result of her measurement will always be $|0\rangle$, which will be correctly interpreted as a '0'. After measurement, the state remains unchanged.*
- *If the receiver chooses to decode $|\chi\rangle$ using the ⊠ basis, the result of her measurement will be either $|+\rangle$ or $|-\rangle$ with equal probability; a $|+\rangle$ would be interpreted as a '0' while a $|-\rangle$ would be interpreted as a '1'. After measurement, the state changes permanently to $|+\rangle$ or $|-\rangle$ respectively.*

The receiver of the message sequence will not know the basis originally chosen to encode each bit, and neither will the eavesdropper. They are both bound to choose incorrect bases for measurement several times. As illustrated in the example above, the measurement not only is likely to yield an incorrect result, but will also change the state of each particle permanently[13]. What the eavesdropper transmits to the receiver is therefore a different state from that originally transmitted.

Example 1.2 *To transmit the sequence '101' the sender decides to use the bases ⊠, ⊞, ⊞ (in that order) to encode each bit to a particle. This means that the particles transmitted have the states $|-\rangle, |0\rangle, |1\rangle$ respectively.*

1. *The eavesdropper does not know the original bases used for encoding, so she chooses at random the bases ⊞, ⊠, ⊞ to decode $|-\rangle, |0\rangle, |1\rangle$. Since the incorrect basis was chosen for the first two particles, the result of measuring them gives a random result, e.g. the state $|0\rangle$ instead of $|-\rangle$ for the first particle, and the state $|+\rangle$ instead of $|0\rangle$ for the second particle. The last particle is measured correctly as a $|1\rangle$ state, and the eavesdropper concludes that the sequence transmitted was '001'. She transmits the sequence of states $|0\rangle, |+\rangle, |1\rangle$ to the receiver.*
2. *The receiver picks the bases ⊠, ⊠, ⊠ to decode the sequence $|0\rangle, |+\rangle, |1\rangle$. The results of the first and last measurements are random, say $|-\rangle$ and $|+\rangle$, and the receiver concludes from her measurements that the sequence transmitted was '100'.*

This example demonstrates how, using conjugate coding, a short bit sequence can be transmitted; although there is an error in the final value received, the eavesdropper certainly has not managed to obtain the original sequence, and that is precisely the achievement of this technique. The errors that result from incorrect choices made by the receiver can be removed through subsequent agreement with the sender; this is the process of *reconciliation* and will be discussed in Chapter 2.

1.3.1 Cryptographic Keys

The use of conjugate coding for cryptographic purposes was proposed by Bennett and Brassard (1984) and Wiedemann (1987). The most important application is *key distribution.* To justify this application, a few words with regard to cryptographic keys are now in order.

In a secret–key cryptosystem, a given message m is encrypted using some algorithm E and a value k, the *key,* which is known only to the sender and receiver. The sender encrypts m using E to form a *cryptogram* c, which is subsequently transmitted:

$$c = E(m, k) \tag{1.6}$$

To recover m from c, a knowledge of k is needed. Only then can the decryption algorithm D be applied:

$$m = D(c, k) \tag{1.7}$$

Note that the key k needs to be established before transmission. If the algorithms E and D are publicly known, the eavesdropper's only task is to obtain the value of k. Generally speaking, the key may be obtained in one of two ways: by attacking sender and receiver directly when it is first established, or by consistently monitoring the cryptogram c to detect patterns or hints suggesting the value of k. The latter of these two approaches is known as *cryptanalysis.*

The problem that needs to be addressed in any secret–key cryptosystem is how to establish the common key k secretly in the first place. This is the so–called *key distribution problem.* In an effort to resolve the key distribution problem, Diffie and Hellman[14] proposed the concept of *public–key cryptography,* where sender and receiver do not need to use a common key. In a public–key cryptosystem, each user possesses a pair of keys (SK, PK) such that $f(SK) = PK$, where f is a one–way function[15]. The *secret key* SK of a particular user is never disclosed and is unique. The *public key* PK is, however, public knowledge.

In order for some user A to send an encrypted message m to user B she must use B's public key:

$$c = E(m, PK_B) \tag{1.8}$$

Only user B can successfully decrypt the cryptogram c, by using her private key SK_B:

$$m = D(c, SK_B) \tag{1.9}$$

In this case, an eavesdropper needs to obtain the secret key SK_B, which is only possible by inverting the function f:

$$SK_B = f^{-1}(PK_B) \tag{1.10}$$

The function f is always carefully chosen, so that its inversion is a computationally intractable task.

Since the potential of quantum computers was discovered, the security of public–key cryp-

tosystems has become debatable. In particular, in popular systems such as RSA, f is a function that multiplies two prime numbers; if Shor's algorithm for efficient quantum factorisation on a quantum computer is implemented, this defense is very vulnerable.

1.3.2 The One-Time Pad

Thus far it has been shown:

- that quantum phenomena, as employed in conjugate coding, can be used to establish arbitrarily secure bit sequences between two parties;
- that secret–key cryptosystems generally suffer from the need to establish keys prior to transmission;
- and that public–key cryptosystems alleviate the need for key distribution, but rely on the intractability of certain computational problems, which makes them vulnerable in the face of quantum computers.

It follows that conjugate coding could likely be used to solve the problem of key distribution. Given that the keys that result from this process are highly secure, the only remaining issue is to select a perfect secret–key cryptosystem that uses them.

A cryptosystem is *perfect* in the information–theoretic sense if, intercepting any cryptogram gives no information whatsoever about the message it encodes. The Vernam cipher, or *one–time pad*, is the classic example of a perfect cryptosystem. In this system, the key used to encode a particular message is at least as long as the message itself, and a different key among these is used for each transmission. Since a different key is used every time, no gain of information can ever be made by eavesdropping on transmissions.

Combined with a perfect cryptosystem such as the one–time pad, quantum key distribution can be used in a communication system with perfect secrecy.

1.4 Other Quantum Protocols

Gilles Brassard and Charles Bennett extended the principles behind quantum key distribution and quantum oblivious transfer to other cryptographic applications[16]. In particular, they proposed protocols for quantum *bit commitment* and quantum *coin–flipping*, to be discussed next. We will turn our attention to these now, and we will also see how the quantum phenomenon of *entanglement* can be employed to construct useful protocols.

1.4.1 Bit Commitment and Coin Flipping

Commitment as a cryptographic term refers to proof that a certain party possesses a particular datum. Although the datum is known only to that party, with commitment she can definitively prove to any other party that she is using *that* datum and cannot alter it. A commitment protocol thus allows party A to commit to a prediction (i.e. a series of bits) without revealing her prediction to B until some time in the future[17].

Several commitment protocols exist[18] but can be subverted using subtle attacks. The *quantum bit commitment* protocol was originally shown to be *provably* secure[19]. Unfortunately, it was eventually proved that unconditionally secure quantum bit commitment is impossible[20].

Coin flipping is another cryptographic problem akin to bit commitment. Using a coin flipping protocol, two parties can flip a coin at a distance and agree on the outcome. However, the outcome of the flip is not determined individually by one of the two parties, since it is assumed they do not trust each other. A coin flipping protocol allows them to agree on the outcome nevertheless.

In coin flipping, no third party is required to determine who the winner and loser are. There has to be a 50% chance of winning a coin flip for both parties, and any attempt to bias the outcome should be immediately detectable.

The traditional solutions to both problems of bit commitment and coin flipping rely on unproven assumptions about computational complexity for their security. Just as cryptosystems that rely on the difficulty of factoring prime numbers are vulnerable against quantum computers, it is likely that existing schemes for the above problems will become defenseless with an increase in computational power. This is exactly why quantum protocols offer an elegant alternative; their security is only really related to our understanding of the physical world.

1.4.2 ENTANGLEMENT-BASED PROTOCOLS

So far, only superposition and the nature of measurement have been presented as distinctive aspects of quantum theory. There is one more feature of the subatomic world which allows the development of effective communication schemes: *quantum entanglement.*

A system of particles may exist in a certain state, which cannot be broken down and expressed in terms of the individual particles' states; this state, which describes the totality of the particles, is an *entangled* state. An example of such a state is

$$\left|\Psi^{-}\right\rangle_{[\mathbf{12}]} = \frac{1}{\sqrt{2}}\left(|0\rangle_{[\mathbf{1}]}\ |1\rangle_{[\mathbf{2}]} - |1\rangle_{[\mathbf{1}]}\ |0\rangle_{[\mathbf{2}]}\right) \tag{1.11}$$

NOTATION 1.4 *The tensor product of two state vectors $|a\rangle$ and $|b\rangle$ is written $|a\rangle \otimes |b\rangle$ or $|a\rangle\ |b\rangle$. This is frequently shortened to $|ab\rangle$ in the literature. More information about the tensor product of quantum states can be found in Rieffel and Polak (2000); Nielsen and Chuang (2000) and in section 2.1.3.*

Equation (**1.11**) describes the state of a system of two particles (labelled **[1]** and **[2]** respectively). The system is in a superposition of the state $|0\rangle_{[\mathbf{1}]}\ |1\rangle_{[\mathbf{2}]}$ (in which particle 1 is state $|0\rangle$ and particle 2 in state $|1\rangle$) and the state $|1\rangle_{[\mathbf{1}]}\ |0\rangle_{[\mathbf{2}]}$ (in which particle 1 is state $|1\rangle$ and particle 2 in state $|0\rangle$). The state cannot be decomposed into a product of individual states of the form $(a\,|0\rangle + b\,|1\rangle)$, i.e.

$$\nexists a_1, b_1, a_2, b_2 : \quad \left|\Psi^{-}\right\rangle_{[\mathbf{12}]} = (a_1\ |0\rangle_{[\mathbf{1}]} + b_1\ |1\rangle_{[\mathbf{1}]})\,(a_2\ |0\rangle_{[\mathbf{2}]} + b_2\ |1\rangle_{[\mathbf{2}]}) \tag{1.12}$$

The existence of entangled states such as (**1.11**) led Einstein and some of his contemporaries to doubt the validity and completeness of quantum theory. The primary reason for this is that entangled quantum states exhibit unusual properties when measured. Two particles that are entangled are correlated in such a way that, if one of the two is measured, the other is affected. By

measuring one particle, the result of measuring the other at any point in the future is known; that is, the second measurement is always deterministic. What is more, this effect occurs *independently* of the physical distance between the two particles; it is also referred to as the phenomenon of *quantum non–locality* for this reason.

Example 1.3 *If particle 1 in the system described by (**1.11**) is measured, and the result is* $|0\rangle$*, then measuring particle 2 subsequently will always yield* $|1\rangle$*. This is indicated by the order of the two states in the expression* $|0\rangle_{[\mathbf{1}]}\ |1\rangle_{[\mathbf{2}]}$*. Similarly, if particle 1 is measured and yields* $|1\rangle$*, measuring particle 2 afterwards will yield* $|0\rangle$*.*

A rewarding discussion of entanglement, with references to the original works on the subject, is to be found in Bub (2002). Interestingly, entanglement has been used as the foundation for several theoretical protocols. Of these, *dense coding* (section 2.4) and *quantum teleportation* (section 2.5) deserve the greatest mention.

1.5 Protocol Specification and Verification

It is by now clear that quantum–mechanical effects may be used effectively to devise powerful communication protocols. It must be borne in mind, however, that these schemes have been, to date, largely the result of research in theoretical physics. This is no surprise, of course, since a thorough grounding in physics is necessary to grasp and apply the phenomena of quantum mechanics to practical situations.

General communication protocols, and especially security protocols, have always been under the scrutiny of computer scientists and numerous techniques for their design and testing have been developed. Process calculi, such as CCS[21], CSP[22] and ACP[23], have all been used for methodical specification of protocols, and for the construction of formal proofs. Automated verification tools and programming languages have been proposed for precisely these tasks. The two principal variants of automated verification, *model checking* and *theorem proving*, are frequently targeted at protocols and have both been used in the past to detect faults and subtle bugs in them. As mentioned in the Preface, the FDR model checker allowed Gavin Lowe to uncover a flaw in the Needham–Schroeder security protocol; the SPIN model checker, intended for testing distributed systems software in general, started life as a tool for checking and correcting the behaviour of telecommunications protocols and telephone switches[24].

So why not consider using formal methods and validation tools to investigate quantum protocols, such as those presented previously? *Indeed, this is the rationale for the current thesis.* The use of modelling software and specification formalisms is likely to produce elegant and unambiguous protocol definitions; it will allow for automated construction of useful proofs; and it will provide potential system implementors with a solid theoretical foundation and a platform for testing design ideas.

1.6 Scope and Contents of the Following Chapters

The emphasis in the remainder of this thesis will be put on the use of model checking to investigate the properties of quantum protocols, e.g. the security of quantum key distribution. Proofs of security already exist, but to understand them one needs to have considerable background in a variety of fields. Using software tools instead is a practical alternative. We will pay attention to techniques for formal specification, including process calculi and quantum programming languages. An imperative specification language devised by the author, qSpec, will be described; its syntax will be given by way of example, and we will present an operational model for programs in this language.

Chapter 2 describes a variety of quantum protocols, expanding the material discussed here. We will review notation and ideas from quantum theory, thus clarifying further the symbols used in the current chapter; we will discuss the concept of security from an information–theoretic point of view; and quantum key distribution protocols BB84, B92 and E91 will be described carefully. Protocols for dense coding and quantum teleportation will also be discussed.

Chapter 3 is concerned with the theory and practice of *model checking*, which is the method with which we propose to analyse quantum protocols. The function and architecture of a model checker will be reviewed, with particular attention to the syntax and semantics of temporal logic. We will show how we can formally specify the measurement of entangled quantum particles using logic, probability, and the prism software tool.

The principal outcome of this work is the use of the prism and spin model checkers as vehicles for verifying the security of quantum key distribution; this is the subject matter of Chapter 4, where it is shown that the probability of a successful eavesdrop diminishes exponentially with the number of transmitted bits.

Alternative approaches to formal specification and verification of quantum protocols are examined in the fifth chapter. We discuss quantum process algebras, including cqp and *QPAlg*; the logic of knowledge and time for quantum systems proposed by Meyden and Patra (2003); software such as QCSim and *probUSM*, and also the probabilistic specification language probmela[25].

In Chapter 5 all these formalisms are discussed, and qSpec is presented in detail; the final chapter points to directions for future work.

2

A Survey of Quantum Protocols and Security Criteria

THE DISCUSSION NOW TURNS TO the protocols used in various quantum communication schemes and, in particular, in quantum cryptography. There are four main classes of quantum protocols with purely cryptographic goals, namely:

- **key distribution**

  This class of protocols includes BB84[1], B92[2], and Ekert's protocol[3].

- **establishing desired levels of privacy**

  This class includes specialised protocols which do not necessarily employ quantum effects, such as the schemes for secret–key reconciliation[4] and privacy amplification[5].

- **commitment**

  There are several protocols for quantum bit commitment[6], but it has been proved that they can never be unconditionally secure[7].

- **oblivious transfer**

  To this category belong variations of Wiesner's protocol for quantum oblivious transfer[8].

There are two more kinds of quantum protocol that are of interest, but these are not intended for cryptographic purposes:

- **dense coding**

  A dense coding protocol[9] allows a quantum channel to be used to efficiently transmit classical bits.

- **teleportation**

  Teleportation protocols[10] allow a quantum state to be exchanged using only a classical channel.

A proper presentation of all these protocols and the issues associated with them necessitates a basic understanding of the formalism of quantum mechanics; section 2.1 is a summary of important results and notation from this area. The exposition given here is based on Cohen-Tannoudji

et al. (1977), which is the most comprehensive reference available on the subject. Since the foremost concern in this thesis is security, a thorough discussion of this concept and Shannon's classic formalisation is given in section 2.2. Following these preliminaries, the protocols for key distribution and establishing desired levels of privacy are described. We will also have a brief look at dense coding and teleportation.

2.1 Elements of Quantum Theory and its Mathematical Formalisation

Quantum mechanics is that part of Physics which deals with phenomena at the atomic scale; its forefathers include such major figures as Albert Einstein, Erwin Schrödinger, and Paul Dirac. It was in the 20th century that these, and many other respected scientists, came to terms with the world of the atom, and how drastically it differs from everyday experience.

The cornerstone of quantum theory is the realisation that objects on an atomic scale behave in a random fashion, and not deterministically as classical physics would lead us to believe. Furthermore, the phenomena at this level involve discrete, or *quantised* changes, as opposed to the familiar continuous quantities in classical physics. Another important result in quantum theory, known as the Heisenberg *uncertainty principle,* places bounds on the amount of information that can ever be extracted through observation or measurement. That is, the amount of accuracy in the measurement of any physical quantity is strictly limited.

Quantum theory challenges the very notion of a particle. It turns out that quantum "objects" can exhibit the key property of a particle, namely locality in space, as well as the interference phenomena of waves. This leads one naturally to conceive of a concise, mathematical description of a quantum system, independently of its characterisation as a "wave" or a "particle"; this is termed the *quantum state.*

A quantum state is a vector which belongs to an abstract vector space. From this vector one can extract everything that is "knowable" about the system it represents (see also section 1.2.1). The vector space is referred to as the *state space* of the system, while the vectors themselves are called "kets". A ket is written as a label enclosed in $|\cdot\rangle$ parentheses. Each ket is a *complex–valued vector,* and the state space has the structure of a *Hilbert space*.

The counterpart to a ket is a "bra", and all bras form the *dual* of the state space. When a bra $\langle\varphi|$ and a ket $|\psi\rangle$ are multiplied, they form the scalar product $(|\varphi\rangle, |\psi\rangle)$:

$$\langle\varphi \,|\, \psi\rangle = (|\varphi\rangle, |\psi\rangle) \tag{2.1}$$

The origin of the terms "bra" and "ket" is now apparent, since these are the two elements of a "bracket" $\langle\varphi \,|\, \psi\rangle$. Note that a bra is a *linear functional:* it maps any ket to a complex number.

To summarise:

- a quantum state is described by an abstract vector,or ket, of the form $|\psi\rangle$;
- all the possible states of a quantum system belong to a Hilbert space $\mathscr{H}$;
- and to each ket $|\psi\rangle$ corresponds a bra $\langle\varphi|$, with the product of any ket and bra being a complex number.

The following definition is of particular utility in some quantum protocols, as will be seen in later sections.

Definition 2.1 *Two states* $|\varphi\rangle$ *and* $|\psi\rangle$ *are said to be* orthogonal *if* $\langle\varphi \,|\, \psi\rangle = 0$.

2.1.1 Bases and Representations

Thus far, the mathematical formalism of quantum mechanics has been presented as a calculus in an abstract vector space. However, quantum states can be expressed in concrete terms; this involves choosing a suitable *representation*.

The state space of a quantum system, just as any vector space, is spanned by a certain number of *basis vectors*. This means that any quantum state can be expressed as a linear combination of these:

$$|\psi\rangle = \sum_i c_i \,|u_i\rangle \tag{2.2}$$

Expression (**2.2**) is the *representation* of state $|\psi\rangle$ in the *basis* $\{|u_i\rangle\}$. A particular basis $\{|u_i\rangle\}$ is *orthonormal* if all basis vectors are orthogonal to each other:

$$\langle u_i \,|\, u_j\rangle = \delta_{ij} \tag{2.3}$$

where δ_{ij} has value 1 when $i = j$ and 0 otherwise.

With the introduction of a basis, every quantum state of a particular system can be expressed concretely; it suffices to obtain the coefficients c_i in equation (**2.2**), which can be written, for each i,

$$c_i = \langle u_i \,|\, \psi\rangle \tag{2.4}$$

A ket is written conventionally as a column vector containing the coefficients c_i:

$$\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

while a bra is written as a row vector, so that the product of a bra and a ket is a number:

$$\langle\varphi \,|\, \psi\rangle = \begin{pmatrix} d_1 & d_2 & \cdots & d_n \end{pmatrix} \times \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = d_1c_1 + d_2c_2 + \cdots + d_nc_n \tag{2.5}$$

Finally, an operator is concretely expressed as a matrix, whose elements A_{ij} satisfy:

$$A_{ij} = \langle u_i \,|\, A \,|\, u_j \rangle \tag{2.6}$$

EXAMPLE 2.1 *Consider a two–state system such as a hydrogen atom; its state belongs to a 2–dimensional Hilbert space $\mathscr{H}_2$, with basis states $|u_1\rangle$ (the atom's ground state) and $|u_2\rangle$ (the atom's excited state). In general, the overall state of the atom when unobserved is*

$$|\psi\rangle = \sum_i c_i \,|u_i\rangle = c_1 \,|u_1\rangle + c_2 \,|u_2\rangle$$

Since $|u_1\rangle$ and $|u_2\rangle$ are basis vectors, to make the basis orthonormal the following must hold:

$$\begin{aligned} \langle u_1 \,|\, u_2 \rangle &= 0 \\ \langle u_2 \,|\, u_1 \rangle &= 0 \end{aligned}$$

The representations of basis vectors $|u_1\rangle$ and $|u_2\rangle$ are

$$|u_1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |u_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

The overall state of the atom can now be written

$$|\psi\rangle = c_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

2.1.2 THE NATURE OF QUANTUM MEASUREMENT

The only means of interacting with the atomic world is through direct observation using appropriate equipment. Observation, or *measurement,* is a process which generally alters any quantum state. The remarkable aspect of quantum measurement is precisely this: it is impossible to create a perfectly selective or discriminatory device for obtaining accurate results about a quantum system. The very act of measurement causes an irreversible disturbance, and this means that the exact quantum state of a system can never be obtained[11].

Measurement of any quantum system can only ever give one of a set of privileged results, namely those associated with basis vectors of the state space. Thus the results of any measurement are limited, or *quantised,* to certain *eigenvalues.*

With each possible measurement result is associated a definite system state, called an *eigenstate.* Each eigenstate corresponds to a basis vector of the state space.

Also, after measurement is performed, the quantum state changes permanently. If the result of a measurement is basis vector u_i, then the quantum state changes to u_i permanently thereafter. Any subsequent measurement with the same basis will always yield u_i.

More interestingly, the outcome of a measurement is subject to the laws of probability. A

quantum state

$$|\psi\rangle = \alpha\,|u_1\rangle + \beta\,|u_2\rangle \tag{2.7}$$

will yield one of the eigenstates $|u_1\rangle$ or $|u_2\rangle$ with respective probability $|\alpha|^2$ or $|\beta|^2$. For this reason, in any linear expansion

$$|\psi\rangle = \sum_i c_i\,|u_i\rangle \tag{2.8}$$

the c_i are referred to as *probability amplitudes*.

Example 2.2 *Consider a system with state*

$$|\psi\rangle = \sqrt{0.3}\cdot|0\rangle + \sqrt{0.7}\cdot|1\rangle$$

Using the notation introduced in Chapter 1, $\boxplus$ *is the basis* $\{|0\rangle, |1\rangle\}$*. A measurement of* $|\psi\rangle$ *using* $\boxplus$ *will yield the new state* $|\psi'\rangle = |0\rangle$ *with probability* $\left|\sqrt{0.3}\right|^2 = 0.3$ *or the new state* $|\psi'\rangle = |1\rangle$ *with probability* $\left|\sqrt{0.7}\right|^2 = 0.7$*. If the basis* $\boxtimes = \{|-\rangle, |+\rangle\}$ *is used instead, however, the result will be either* $|-\rangle$ *or* $|+\rangle$ *at random, with different respective probabilities.*

2.1.3 The State of a Composite Quantum System

Having understood how a single quantum system may be described by means of a state vector, the reader is prepared to consider how to describe *aggregates* of such systems. In particular, if S_1 and S_2 are two isolated physical systems with state spaces $\mathscr{E}_{[1]}$ and $\mathscr{E}_{[2]}$, how can one describe the combined system $S_1 + S_2$? This is the final aspect of the formalism of quantum mechanics that needs to be dealt with here.

Given two quantum systems whose state spaces are known, the state space that is formed when the two are combined is called the tensor product space. If $\mathscr{E}_{[1]}$ and $\mathscr{E}_{[2]}$ denote the state spaces of the first and second system respectively, the tensor product space is written

$$\mathscr{E} = \mathscr{E}_{[1]} \otimes \mathscr{E}_{[2]} \tag{2.9}$$

and describes the totality of states in which the two systems may be found *together.* The tensor product space $\mathscr{E}$ consists of vectors which are *linear combinations* of products of the form

$$|\varphi\rangle_{[1]} \otimes |\chi\rangle_{[2]} \tag{2.10}$$

where $|\varphi\rangle_{[1]}$ denotes the individual state of system S_1, and $|\chi\rangle_{[2]}$ denotes the individual state of system S_2. Thus, if the composite system $S_1 + S_2$ is found to be in state (**2.10**), then its components will be in states $|\varphi\rangle$ and $|\chi\rangle$ respectively.

A vector of the form (**2.10**) is the *tensor product* of vectors $|\varphi\rangle_{[1]} \in \mathscr{E}_{[1]}$ and $|\chi\rangle_{[2]} \in \mathscr{E}_{[2]}$ and possesses the following properties:

i. it is *linear* with respect to multiplication by complex constants, e.g.

$$\left[\lambda \cdot |\varphi\rangle_{[1]}\right] \otimes |\chi\rangle_{[2]} = \lambda \cdot \left[|\varphi\rangle_{[1]} \otimes |\chi\rangle_{[2]}\right]$$

ii. it is *distributive* with respect to vector addition, e.g.

$$|\varphi\rangle_{[1]} \otimes \left[|\chi\rangle_{[2]} + |\chi'\rangle_{[2]}\right] = |\varphi\rangle_{[1]} \otimes |\chi\rangle_{[2]} + |\varphi\rangle_{[1]} \otimes |\chi'\rangle_{[2]}$$

iii. If $|u_i\rangle_{[1]}$ is a basis of the state space $\mathscr{E}_{[1]}$ and $|v_m\rangle_{[2]}$ is a basis of $\mathscr{E}_{[2]}$, then the set of vectors

$$\left\{|u_i\rangle_{[1]} \otimes |v_m\rangle_{[2]}\right\}$$

is a basis of $\mathscr{E}$.

The vectors in $\mathscr{E}$ take two forms; on one hand, there are vectors which can be expanded into a product of vectors of $\mathscr{E}_{[1]}$ and $\mathscr{E}_{[2]}$. Thus, if

$$\begin{aligned} |\varphi\rangle_{[1]} &= a_1 \cdot |u_1\rangle + a_2 \cdot |u_2\rangle \\ \text{and } |\chi\rangle_{[2]} &= b_1 \cdot |v_1\rangle + b_2 \cdot |v_2\rangle \end{aligned}$$

then an example of a state which is *decomposable* is

$$\begin{aligned} |\psi\rangle_{[12]} &= a_1 \cdot b_1 \cdot |u_1 v_1\rangle + a_1 \cdot b_2 \cdot |u_1 v_2\rangle + a_2 \cdot b_1 \cdot |u_2 v_1\rangle + a_2 \cdot b_2 \cdot |u_2 v_2\rangle \\ &= (a_1 \cdot |u_1\rangle + a_2 \cdot |u_2\rangle) \otimes (b_1 \cdot |v_1\rangle + b_2 \cdot |v_2\rangle) \\ &= |\varphi\rangle_{[1]} \otimes |\chi\rangle_{[2]} \end{aligned}$$

On the other hand, there exist state vectors which cannot be expressed as products of vectors in $\mathscr{E}_{[1]}$ with vectors in $\mathscr{E}_{[2]}$. These are called *entangled states*:

$$|\psi'\rangle_{[12]} = c_1 \cdot |u_1 v_1\rangle + c_2 \cdot |u_2 v_2\rangle = c_1 \cdot |u_1\rangle_{[1]} \otimes |v_1\rangle_{[2]} + c_2 \cdot |u_2\rangle_{[1]} \otimes |v_2\rangle_{[2]} \tag{2.11}$$

EXAMPLE 2.3 *Expression (**2.11**) describes the so–called* EPR state *(due to Einstein, Podolsky and Rosen) when $c_1 = c_2 = \frac{1}{\sqrt{2}}$ and $|0\rangle$ replaces $|u_1\rangle$ and $|v_1\rangle$, and $|1\rangle$ replaces $|u_2\rangle$ and $|v_2\rangle$). It is thus written as:*

$$|\psi'\rangle_{[12]} = c_1 \cdot |00\rangle + c_2 \cdot |11\rangle = c_1 \cdot |0\rangle_{[1]} \otimes |0\rangle_{[2]} + c_2 \cdot |1\rangle_{[1]} \otimes |1\rangle_{[2]} \tag{2.12}$$

It is indeed entangled, *since there do not exist any constant numbers a_1, a_2, b_1, b_2 such that*

$$|\psi'\rangle_{[12]} = (a_1 \cdot |0\rangle + a_2 \cdot |1\rangle) \otimes (b_1 \cdot |0\rangle + b_2 \cdot |1\rangle)$$

A product state such as (**2.10**) is considered to represent the simple juxtaposition of two systems; the results of the two types of possible measurements, bearing either on one system or on the other, correspond to independent random variables. A state which is not a tensor product describes a situation in which there are *correlations* between the systems involved. A measurement

on part of such a system will affect both component states; the result can no longer be expressed as a change in either one of the two component states.

EXAMPLE 2.4 *Consider the possible results of measuring the* EPR *state (***2.11***). The observer tries to observe system S_1 first, then system S_2. From the definition of the state*

$$|\psi'\rangle_{[12]} = c_1 \cdot |00\rangle + c_2 \cdot |11\rangle = c_1 \cdot |0\rangle_{[1]} \otimes |0\rangle_{[2]} + c_2 \cdot |1\rangle_{[1]} \otimes |1\rangle_{[2]}$$

it can be seen that the result of the measurement will be $|0\rangle_{[1]} \otimes |0\rangle_{[2]}$ with probability c_1^2, or $|1\rangle_{[1]} \otimes |1\rangle_{[2]}$ with probability c_2^2. The first system will either be found to be in state $|0\rangle$ or in state $|1\rangle$. If it is found in state $|0\rangle$, then subsequent measurement of the second system will yield $|0\rangle$ with certainty. If it is found in state $|1\rangle$, then subsequent measurement of the second system will yield $|1\rangle$ with certainty.

2.2 PERFECT SECRECY AND SECURITY

Next, a handful of concepts from Shannon's theory of information will be presented. Of interest to us is the application of information theory to cryptographic systems[12]. Formal definitions of the notion of security will also be given.

As explained at length in Chapter 1, the objective of a cryptographic protocol is to code messages in such a manner as to prevent an enemy, such as an eavesdropper, from learning their content. The situation is described by two sets of probabilities; the so–called *a priori* probabilities quantify the likelihood of a particular message and key being chosen in the first instance; they represent the enemy's *a priori* knowledge. When a cryptogram is intercepted, the enemy can compute *a posteriori* probabilities of key and message combinations.

The set of possible messages is termed the *message space* $\{m_i\}$. The probability that message m_i is chosen for transmission is written $P(M) = \Pr\{M = m_i\}$, where M is a random variable over the message space. One can similarly define the probability that a key k_i is chosen as $P(K) = \Pr\{K = k_i\}$, and the probability $P(E) = \Pr\{E = e_i\}$ that cryptogram e_i is intercepted. Keys are chosen from a key space $\{k_i\}$ and cryptograms from a space $\{e_i\}$, while K and E are random variables over these spaces.

The *a priori* probability $P(E \mid M) = \Pr\{E = e_i \mid M = m_i\}$ determines with what likelihood the cryptogram e_i is produced when message m_i is to be transmitted. On the other hand, when e_i is intercepted by an enemy, she can compute with what probability m_i was the chosen message; this latter is the *a posteriori* probability of message m_i "given cryptogram e_i" and is written $P(M \mid E) = \Pr\{M = m_i \mid E = e_i\}$.

Note that the enemy is assumed to know the cryptographic mechanism being used by the sender[13], as well as the probabilities $P(K)$ of choosing various keys. The a priori and a posteriori probabilities are related quite simply through Bayes' theorem:

$$P(M \mid E) = \frac{P(M) \cdot P(E \mid M)}{P(E)} \tag{2.13}$$

Shannon also developed the theoretical notion of perfect secrecy:

DEFINITION 2.2 *A cryptosystem is said to have perfect secrecy if, for all cryptograms E, the a posteriori*

probabilities are equal to the a priori probabilities, whatever their value:

$$P(E \mid M) = P(E)$$
$$\text{and conversely } P(M \mid E) = P(M)$$

Perfect secrecy is achieved only in a system where intercepting a cryptogram gives absolutely no information about the content of the message it represents.

The most basic concept in information theory is *entropy*, which quantifies the amount of uncertainty associated with any datum or, put otherwise, the amount of information generated when a datum is first produced. In a cryptographic system there is uncertainty associated with the choice of message,

$$H(M) = -\sum P(M) \log P(M) \tag{2.14}$$

as well as uncertainty associated with the choice of key:

$$H(K) = -\sum P(K) \log P(K) \tag{2.15}$$

When the concept of entropy is applied within the context of a cryptosystem, a useful measure of the system's secrecy results; this is termed the equivocation, given by

$$H(K \mid E) = \sum_{E,K} P(E,K) \cdot \log P(K \mid E) \qquad \text{for keys} \tag{2.16}$$
$$H(M \mid E) = \sum_{E,M} P(E,M) \cdot \log P(M \mid E) \qquad \text{for messages} \tag{2.17}$$

where $P(E,K)$ is the probability that key K and cryptogram E occur together; $P(E,M)$ is similarly defined.

When considering any cryptosystem, one associates with it a formal level of security. The strongest possible definition of security is based on Shannon's ideas:

Definition 2.3 (Unconditional Security) *A cryptosystem with messages M, keys K, and cryptograms E is* perfectly or unconditionally secure *if an enemy with unlimited computational power can learn nothing about a message m_i given the matching cryptogram e_i. This requires that:*

1. *the key used should be at least as long as the message, i.e. $|k_i| \geqslant |m_i|$.*
2. *the same key is never used twice, i.e. every key is used with equal probability and, for all messages m_i and cryptograms e_i, there is a unique key k_i that matches m_i to e_i.*

A cryptosystem rarely achieves perfect security; the most well–known example of a perfectly secure system is the one–time pad, described previously in section 1.3.2. There exist weaker definitions of security, which ascribe limited computational power to the enemy[14]:

Definition 2.4 (Computational Security) *A cryptosystem is* computationally secure *if the best known algorithm for breaking it requires an unreasonably large amount of computational resources.*

Definition 2.5 (Provable Security) *A cryptosystem is termed* provably secure *if breaking it reduces to solving some well–studied hard computational problem.*

We will now proceed to discuss the three main protocols that implement quantum key distribution.

2.3 Quantum Key Distribution in Detail

The most important and useful quantum protocols for cryptography deal with the problem of key distribution, discussed previously in section 1.3.2. Although there exist classical cryptographic schemes which are unconditionally secure, *classical* key distribution with unconditional security is impossible[15].

In classical key distribution, the common key is first established by a single user, and then *distributed* (hence the term) to the other users. In quantum key distribution systems the common key is *generated* through the collaboration of all the users involved. For this reason, quantum key *distribution* is a misnomer[16]; however, it is the preferred term in most of the relevant literature for historical reasons.

Any key distribution protocol generally has to ensure that the key produced is truly random. This implies that an enemy cannot predict its value prior to the protocol, and can only attempt to monitor the legitimate users' communications. Clearly, a key distribution protocol must prevent an enemy eavesdropper from learning any significant part of the key, while also giving legitimate users certainty that they indeed share a *common* key. Any discrepancy between the users' individual keys makes subsequent communication problematic, if not impossible.

Quantum key distribution protocols require quantum channels which, unlike their classical counterparts, cannot be passively monitored. Although, for example, a telephone line can be wire–tapped without this being noticed, any gain of information from a quantum channel necessarily produces a disturbance.

There are three implementations of quantum key distribution, each of which makes use of a different feature of quantum theory:

- **The original BB84 protocol**[17]**,** which uses two conjugate bases (see section 1.2) for encoding bits into sequences of polarised photons;
- **The B92 protocol**[18]**,** which is a simplification of BB84; it uses only two nonorthogonal quantum states to encode information;
- **The Ekert or E91 protocol**[19]**,** which uses entangled pairs of particles to establish a common bit sequence and a so–called Bell inequality to detect eavesdropping.

Common to these variations is the underlying conceptual procedure, which involves four basic steps[20], shown in Figure 2.3.

To implement this procedure, the following quantum–mechanical features are used:

- An unknown quantum state cannot be cloned;
- Non–orthogonal states are generally impossible to distinguish with 100% accuracy;

Figure 2.1 The conceptual procedure underlying all quantum key distribution protocols.

1. A and B independently generate private, random binary sequences $\{x_m\}$ and $\{y_n\}$ respectively, of length $m \gg n$.
2. A prepares a sequence of m tokens (which are actual quantum states) representing the binary sequence $\{x_m\}$; one kind of token is used to represent a binary 0, and a different kind is used to represent a binary 1. She sends the tokens to B.
3. B decodes the tokens using the same convention, in order to recover $\{x_m\}$. He reports to A, for which i, $y_i = x_i$. A subset of these common bits is chosen and forms the *tentative* key.
4. The tentative key is processed further, in order to correct errors and to eliminate any information which may have leaked to an eavesdropper. This results in a *final key.*

- Any measurement of states, with high probability, changes them irreversibly — leading to a significant and detectable rate of errors.

These features are exploited in quantum key distribution protocols so as to prevent an eavesdropper, with arbitrarily high probability, from learning the key. Any eavesdropping attempt can be detected due to the disturbance caused by measurement.

The three main protocols, BB84, B92 and E91 are variations on the procedure in Figure 2.3. We will now discuss each protocol in detail.

2.3.1 THE BB84 PROTOCOL

In BB84, for A and B to establish a secret key, A encodes a sequence of bits as a set of polarised photons. BB84 is a conjugate coding scheme, involving the use of the bases $\boxplus$ and $\boxtimes$. Each bit is represented by a photon in one of the following quantum states:

$$|\Psi(0,\boxplus)\rangle = |0\rangle \tag{2.18}$$
$$|\Psi(1,\boxplus)\rangle = |1\rangle \tag{2.19}$$
$$|\Psi(0,\boxtimes)\rangle = |+\rangle \tag{2.20}$$
$$|\Psi(1,\boxtimes)\rangle = |-\rangle \tag{2.21}$$

NOTATION 2.1 (MAYERS (2001)) *The symbol $|\Psi(d,b)\rangle$ represents the quantum state representing bit d when basis b is used. For instance, $|\Psi(d,\boxplus)\rangle$ denotes a rectilinearly polarised photon that represents bit d; when $d = 0$ this is a horizontally polarised photon (polarisation angle 0°), when $d = 1$ it is a vertically polarised photon (polarisation angle 90°).*

Let's consider, by way of example, the steps taken by two parties using BB84. Example 2.5 demonstrates the behaviour of the sender of a message, conventionally called *Alice;* example 2.6 shows the steps taken by the receiver, conventionally named *Bob.*

EXAMPLE 2.5 *Suppose Alice wishes to transmit the binary sequence $\{d_i\}$ where*

i	*1*	*2*	*3*	*4*	*5*	*6*	*7*
d_i	1	0	1	1	1	0	0

to Bob using the BB84 coding scheme. For each bit d_i, she chooses an encoding basis at random. Say for example, that her choice of basis for the i-th bit is denoted by b_i, and the sequence of all these bases is

i	1	2	3	4	5	6	7
b_i	⊞	⊠	⊞	⊠	⊞	⊠	⊞

Alice prepares the sequence of states $\{|\Psi(d_i, b_i)\rangle\}$:

i	1	2	3	4	5	6	7
$\|\Psi(d_i, b_i)\rangle$	$\|\Psi(1,⊞)\rangle$	$\|\Psi(0,⊠)\rangle$	$\|\Psi(1,⊞)\rangle$	$\|\Psi(1,⊠)\rangle$	$\|\Psi(1,⊞)\rangle$	$\|\Psi(0,⊠)\rangle$	$\|\Psi(0,⊞)\rangle$

Then she transmits, on a noiseless, or perfect quantum channel, to Bob the corresponding sequence of photons:

i	1	2	3	4	5	6	7
$\|\Psi(d_i, b_i)\rangle$	$\|1\rangle$	$\|+\rangle$	$\|1\rangle$	$\|-\rangle$	$\|1\rangle$	$\|+\rangle$	$\|0\rangle$
Photon Polarisation Angle	90°	45°	90°	135°	90°	45°	0°

Bob must make appropriate measurements on the photons to recover $\{d_i\}$.

Notation 2.2 *We will use the conventional names "Alice" and "Bob" to refer to the sender and the receiver respectively. "Alice" and the abbreviation A will be used* ***interchangeably****. "Bob" and the abbreviation B will be used interchangeably.*

To summarise: after representing an initial binary sequence using conjugately coded photons, Alice transmits the photons to Bob through a quantum channel. Then, Bob attempts to reconstruct the binary sequence by making measurements on the photons.

In order to measure a photon correctly, Bob must use the same basis as that Alice used for encoding; so, in order to extract bit d from $|\Psi(d, b)\rangle$, Bob must choose b as a decoding basis. If he chooses the wrong basis, he will obtain a random result (as required by quantum theory).

Example 2.6 *Bob receives the sequence*

i	1	2	3	4	5	6	7
$\|\Psi(d_i, b_i)\rangle$	$\|1\rangle$	$\|+\rangle$	$\|1\rangle$	$\|-\rangle$	$\|1\rangle$	$\|+\rangle$	$\|0\rangle$
Photon Polarisation Angle	90°	45°	90°	135°	90°	45°	0°

He does not know what bases b_i were used for encoding, so he chooses a basis b'_i at random with which to decode each d_i. His choices of b'_i are, say,

i	1	2	3	4	5	6	7
b'_i	⊞	⊞	⊠	⊠	⊞	⊞	⊠

His choices for $i = 1, 4, 5$ match Alice's. Thus his measurements on these photons will yield the correct result, i.e. measuring $|\Psi(d_1, b_1)\rangle$ with $b'_1 = b_1 = ⊞$ will give $d_1 = 1$; Bob will also obtain d_4 and d_5

correctly. For the other photons $b'_i \neq b_i$ so the results of the corresponding measurements will be random; in some cases the result $d'_i = d_i$, while in other cases the converse is true. The results of Bob's measurements in these cases are, say, $d'_2 = 1 - d_2 = 1$; $d'_3 = d_3 = 1$; $d'_6 = d_6 = 0$; $d'_7 = 1 - d_7 = 1$. To summarise, the sequence that Bob obtains by making measurements are, in this example,

i	1	2	3	4	5	6	7
d'_i	1	1	1	1	1	0	1

which differs from $\{d_i\}$ in 4 positions. Now, Bob makes a telephone call to Alice and tells her his choices of b'_i; she then tells Bob, for each i, whether his choice was correct. In those cases where Bob's choices differ from Alice's (i.e. for $i = 2, 3, 6$ and 7) the corresponding bits are discarded. The final, common key that is established is $k = d_1 d_4 d_5 = d'_1 d'_4 d'_5 = 111$.

A detailed description of the steps in the BB84 protocol is given in Figure 2.2. We assume that no eavesdropper is actually present, as we have done in the examples.

When an eavesdropper is present, it is made manifest to users A and B; the eavesdropper introduces additional errors as a consequence of not knowing which basis to decode with. The eavesdropper sees each photon $|\Psi(d_i, b_i)\rangle$ and tries to measure it with a randomly chosen basis $\tilde{b}_i$. Only if $\tilde{b}_i = b_i$ does the photon remain unchanged. But, just as for Bob, this can only occur with probability 0.5; with equal probability, there will be cases in which $\tilde{b}_i \neq b_i$. Whenever the eavesdropper chooses the wrong basis for a particular photon, that photon will be disturbed irreversibly. This allows Alice and Bob to detect the eavesdropper's presence.

If the quantum channel is error–free, i.e. noiseless, then using the same basis to measure a photon as that used to encode a value on it will yield the correct result with certainty. That is to say, for a noiseless channel:

$$(b'_i = b_i) \Rightarrow (d'_i = d_i) \tag{2.22}$$

If, for some i, the implication (**2.22**) does not hold, this means that an eavesdropper has disturbed the quantum state $|\Psi(d_i, b_i)\rangle$. But, to reiterate, this is only true for noiseless channels.

EXAMPLE 2.7 *To make this clear, consider again the transmissions in Examples 2.5 and 2.6. Say that, when $|\Psi(d_1, b_1)\rangle = |\Psi(1, \boxplus)\rangle = |1\rangle$ is transmitted, the eavesdropper attempts to measure it using basis $\boxtimes$. The eavesdropper will obtain one of the results $|+\rangle$ and $|-\rangle$; after this, the photon's state changes permanently to $|+\rangle$ or $|-\rangle$ respectively. Thus, $|\Psi(1, \boxplus)\rangle$ is replaced by $|\Psi'(x, \boxtimes)\rangle$ where x is 0 or 1, at random. Now, what Bob receives is $|\Psi(d_1, b_1)\rangle = |\Psi'(x, \boxtimes)\rangle$. In the example, Bob's choice of basis for $i = 1$ is $b'_1 = \boxplus$. Using this basis for measurement, he will obtain a random result. When he discloses to Alice that the rectilinear basis was used, she will inform him that $b'_1 = b_1$; however, it will be found that $d'_1 \neq d_1$. This error is due to the disturbance caused by the eavesdropper.*

On a realistic, noisy quantum channel, photons are likely to be disturbed by the channel itself, independently of whether an eavesdropper is present or not. So there will be bit positions i for which, although $b'_i = b_i$, we will have $d'_i \neq d_i$. These errors must be detected, so that the two users do eventually obtain a common bit sequence. Therefore, an additional step, a test for errors, is normally included in BB84; it is performed in 2(b) (see Figure 2.2).

Figure 2.2 The BB84 Protocol. The quotations are from the original paper by Bennett and Brassard (1984).

1. Transmissions over a Quantum Channel

" ... one user ('Alice') chooses a random bit string and a random sequence of polarization bases (rectilinear or diagonal). She then sends the other user (Bob) a train of photons, each representing one bit of the string in the basis chosen for that bit position, a horizontal or 45–degree photon standing for a binary zero and a vertical or 135–degree standing for a binary 1. "

(a) A generates a private, random binary sequence $\{d_i\}$ of length m.

(b) A prepares a sequence $|\Psi(d_i, b_i)\rangle$ of m polarised photons, each representing a bit from $\{d_i\}$; for each photon she selects a basis at random, i.e. $b_i = \boxplus$ or $b_i = \boxtimes$ with equal probability. The possible values of $|\Psi(d_i, b_i)\rangle$ are given by equations (**2.18**) to (**2.21**).

" As Bob receives the photons he decides, randomly for each photon and independently of Alice, whether to measure the photon's rectilinear polarization or its diagonal polarization, and interprets the result as a zero or one. As explained [before] ... a random answer is produced and all information lost when one attempts to measure the rectilinear polarization of a diagonal photon, and vice versa. "

(c) B receives the photons and chooses a basis b'_i at random with which to measure each one. The results of B's measurements form a sequence of bits $\{d'_i\}$.

2. Transmissions over a Classical, Possibly Public Channel

" Subsequent steps of the protocol over an ordinary public communications channel, assumed to be susceptible to eavesdropping but not to the injection or alteration of messages. Bob and Alice first determine, by public exchange of messages, which photons were successfully received and of these which were received with the correct basis. If the quantum transmission has been undisturbed, Alice and Bob should agree on the bits encoded by the photons, even this data [sic] has never been discussed over the public channel. "

(a) B reports to A, for each i, the value of b'_i. A then tells him whether $b'_i = b_i$; if so, A and B know with certainty that, for that particular value of i, $d'_i = d_i$. All values of d_i (respectively, d'_i) for which the same basis has been used by A and B are kept and form the *tentative key*; other values are discarded.

(b) The tentative key is processed further, in order to correct errors (i.e. cases in which $b'_i = b_i$ but $d'_i \neq d_i$ due to channel faults) and to eliminate any information which may have "leaked" about the tentative key. This results in a *final key.*

The test for errors also serves as a test for eavesdropping; this is because the presence of an eavesdropper results in additional errors. But how do the two legitimate users A and B distinguish channel–induced errors from errors due to eavesdropping?

Actually, A and B have to agree on a tolerated error rate, ε_T; this can be broken down as follows:

$$\varepsilon_T = \left\lceil \frac{n_e}{N} \right\rceil \times 100\% = \left\lceil \frac{n_{e,ch} + n_{e,v}}{N} \right\rceil \times 100\%$$

where $n_{e,ch}$ is the number of channel errors and $n_{e,v}$ is the number of errors due to eavesdropping, when N particles are transmitted in total. If the actual error rate, $\varepsilon = \left\lceil \frac{n_e}{N} \right\rceil$, for a particular transmission is below ε_T, it is assumed that the errors are solely due to channel faults. Otherwise, the presence of an eavesdropper is suspected ($\varepsilon_T \gg \varepsilon_{ch}$) and the protocol is aborted. Note that, if A and B share some secret information prior to the protocol, they can continue with the protocol

and establish a truly secret key, even in the presence of an eavesdropper[21].

EXAMPLE 2.8 (ERRORS DUE TO EAVESDROPPING) *A transmits the state*

$$|\Psi(1,\boxplus)\rangle = |1\rangle$$

If there are no errors on the channel and no eavesdropping ($\varepsilon = 0$), the bit value corresponding to a measurement by user B using the correct basis ($\boxplus$), which we write as $\mathsf{Meas}_{\boxplus}|\Psi(1,\boxplus)\rangle$, should give the bit originally encoded:

$$\mathsf{Meas}_{\boxplus}|\Psi(1,\boxplus)\rangle = 1$$

However, if there was an intervening eavesdropper (so that $\varepsilon_v \geqslant 0$) who attempted to measure $|\Psi(1,\boxplus)\rangle$ with the incorrect basis, $\boxtimes$, then $|\Psi(1,\boxplus)\rangle$ would change from $|1\rangle$ to either $|+\rangle$ or $|-\rangle$ permanently with equal probability, and the eavesdropper would get a random outcome. Subsequent measurement by B with the correct basis would change $|\Psi(1,\boxplus)\rangle$ to one of $|0\rangle$ or $|1\rangle$ at random. ***So, although B chooses the correct basis, he is only likely to get the correct answer with 50% probability due to the eavesdropper's error.***

EXAMPLE 2.9 (ERROR RATE) *Suppose A and B agree to exchange $N = 30$ photons in total over the quantum channel, and that they define $\varepsilon_T = 5\%$ as the tolerated error rate. This means that only*

$$n_e = \lceil 5\% \cdot 30 \rceil = 2$$

errors are acceptable if the protocol is to succeed; any more than 2 errors and the protocol must be aborted on suspicion of eavesdropping. Of course, in practice, A and B would choose the value of ε_T based on their knowledge of the channel's capabilities. If the channel is likely to induce, say, at least 6 errors, then ε_T should be at the very least $\frac{6}{30} \times 100\% = 20\%$.

The various possible outcomes of user B's measurements are enumerated in table 2.1, for the cases in which photon state $|\Psi(1,\boxplus)\rangle$ is transmitted, and the eavesdropper uses the incorrect (i.e. diagonal) basis. The symbol $\tilde{d}_i$ is used to denote the outcome of the eavesdropper's measurement using basis $\tilde{b}_i$.

Table 2.1 The different cases that arise when an eavesdropper uses the wrong basis to measure a photon, and sends the result to user *B*.

$\tilde{b}_i$	$\tilde{d}_i$	b'_i	d'_i	*Eavesdropper's result*	*B's Result*	*Eavesdropper detected?*
$\boxtimes$	0	$\boxplus$	0	incorrect	incorrect	yes
$\boxtimes$	0	$\boxplus$	1	incorrect	correct	no
$\boxtimes$	0	$\boxtimes$	0	incorrect	incorrect	no; bit is discarded
$\boxtimes$	1	$\boxplus$	0	correct	incorrect	yes
$\boxtimes$	1	$\boxplus$	1	correct	correct	no
$\boxtimes$	1	$\boxtimes$	1	correct	correct	no; bit is discarded

In those cases where user B chooses the same basis as user A, but gets an incorrect result, the presence of an eavesdropper is detected. Essentially, the outcome of B's measurement is conditional on the eavesdropper's choice of basis.

Now, in the second part of BB84, A and B use a classical (i.e. non–quantum) channel. Using this channel, they discuss their choices of bases and discard those bit positions for which they do not match. An eavesdropper can easily listen to their discussion by wire–tapping the classical channel; but he will not gain any information that could allow him to learn the agreed value of the key. Even if he learns the correct bases needed for measurement, it is too late to perform his measurements again — they are irreversible. This assumes, of course, that quantum states cannot be stored in a temporary memory.

It is important to remember that the design of BB84 assumes the following:

1. The classical, potentially public channel handles messages in such a way that they *can* be monitored, but *not* altered or suppressed by an eavesdropper.

2. Quantum transmissions *can* be suppressed or altered but *cannot*, in principle, be monitored without causing a disturbance.

In the final stage of BB84, as for the other schemes for quantum key distribution, two sub–protocols are executed: *secret–key reconciliation* and *privacy amplification*. These will be explained later.

2.3.2 The B92 Protocol

BB84 is not the simplest quantum key distribution protocol. It actually involves using four different quantum states, which are distinguishable from one another with respect to the basis that is used for measurement. BB84 is, in essence, a direct application of Wiesner's ideas on conjugate coding and it depends on the pairwise orthogonality, so to speak, of the states involved; it uses states which are mutually orthogonal to represent '0' and '1' (*cf.* section 2.1 for inner product $\langle a|b\rangle$ of state vectors):

$$\begin{aligned}\langle \Psi(0,\boxplus) \,|\, \Psi(1,\boxplus)\rangle &= \langle 0|1\rangle = 0\\ \langle \Psi(0,\boxtimes) \,|\, \Psi(1,\boxtimes)\rangle &= \langle +|-\rangle = 0\end{aligned}$$

Two years after BB84 was published, Charles Bennett pointed out that, to distinguish between the state representing '0' and the state representing '1', it is unnecessary for the states to be orthogonal; actually, it is not necessary to use conjugate bases at all. The same effects can be produced by using two non–orthogonal states, $|\Psi(0)\rangle$ and $|\Psi(1)\rangle$, to represent '0' and '1'. In order to measure a photon in state $|\Psi(0)\rangle$ (which corresponds to a polarisation of, say 45°) correctly, the polarising filter's axis must be parallel to the photon's polarisation vector (i.e. the polarising filter's axis must also be at 45°). Any other angle will produce a random measurement result. Bennett proposed a variant of the BB84 protocol based on this idea, and it is known as B92.

More formally, B92 uses two non–orthogonal states to represent '0' and '1',

$$\langle \Psi(0) \,|\, \Psi(1)\rangle \neq 0$$

To measure $|\Psi(0)\rangle$ correctly, one must apply the projection operator

$$P_0 = 1 - |\Psi(1)\rangle \langle\Psi(1)| \tag{2.23}$$

whereas, to measure $|\Psi(1)\rangle$ correctly, the projection operator

$$P_1 = 1 - |\Psi(0)\rangle \langle\Psi(0)| \tag{2.24}$$

must be applied.

When applied to $|\Psi(0)\rangle$ the P_0 operator produces a positive value; if it is applied to $|\Psi(1)\rangle$, the state is annihilated. The reverse is true of operator P_1.

EXAMPLE 2.10 *Let $|\Psi(0)\rangle$ and $|\Psi(1)\rangle$ be non-orthogonal state vectors of unit length. What is the outcome of applying, projector P_0 to $|\Psi(1)\rangle$, and projector P_1 to $|\Psi(0)\rangle$?*

We have:

$$\begin{aligned} P_0 |\Psi(1)\rangle &= (1 - |\Psi(1)\rangle \langle\Psi(1)|) \cdot |\Psi(1)\rangle = |\Psi(1)\rangle - |\Psi(1)\rangle \overbrace{\langle\Psi(1) \,|\, \Psi(1)\rangle}^{1} \\ &= |\Psi(1)\rangle - |\Psi(1)\rangle = \vec{0} \\ P_1 |\Psi(0)\rangle &= |\Psi(0)\rangle - |\Psi(0)\rangle \underbrace{\langle\Psi(0) \,|\, \Psi(0)\rangle}_{1} = \vec{0} \end{aligned}$$

B92 can be seen as a simplification of BB84, whose operation depends on the *correct choice of operator,* rather than basis, for measurement. Otherwise, it differs little from the latter. The protocol is listed fully in Figure 2.3.

2.3.3 THE E91 PROTOCOL

The E91 protocol is yet another implementation of quantum key distribution; it involves the use of entangled pairs of particles. It is so named after Artur Ekert, who published it[22] in 1991. The basic setup is the following: users A and B are physically separated and can communicate directly only via classical means. Both A and B independently are linked to a source of entangled spin–$\frac{1}{2}$ particles, so that the first particle of each pair is sent to A and the second particle to B. As we have explained at length previously, entangled particles do not have a known individual state; it is only when they are measured that each particle is assigned an individual state. Now, when A performs a measurement on her particle, this will affect B's particle so that the outcome of B's measurement is known.

EXAMPLE 2.11 *The source generates an entangled pair of particles in the joint state*

$$|\psi_s\rangle_{[\mathbf{12}]} = \frac{1}{\sqrt{2}} (|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle) = \frac{1}{\sqrt{2}} \left(|\uparrow\rangle_{[\mathbf{1}]} |\downarrow\rangle_{[\mathbf{2}]} - |\uparrow\rangle_{[\mathbf{2}]} |\downarrow\rangle_{[\mathbf{1}]} \right)$$

Note that the possible values of spin, when a spin–$\frac{1}{2}$ particle is measured, are $+\frac{\hbar}{2}$ (called "spin–up," corresponding to state $|\uparrow\rangle$) and $-\frac{\hbar}{2}$ (called "spin–down," corresponding to state $|\downarrow\rangle$). If A is given particle 1, and B is given particle 2, then:

Figure 2.3 The B92 Protocol.

1. Transmissions over a Quantum Channel

   (a) A generates a private, random binary sequence $\{d_i\}$ of length m.

   (b) A prepares a sequence $|\Psi(d_i)\rangle$ of m polarised photons, each representing a bit from $\{d_i\}$, with $|\Psi(0)\rangle = |0\rangle$ and $|\Psi(1)\rangle = |-\rangle$, so that $\langle\Psi(0)\,|\,\Psi(1)\rangle \neq 0$.

   (c) B receives the photons and chooses an operator, P_0 or P_1 at random, with which to measure each one. The operators are defined in Equations (**2.23**) and (**2.24**). The choices of operators for all i form a sequence

   $$\{op(i)|op(i) = P_0 \text{ or } op(i) = P_1 \text{ for all } i\}$$

   while the results of B's measurements form a sequence of bits $\{d_i'\}$.

2. Transmissions over a Classical, Possibly Public Channel

   (a) B reports to A, for each i, the choice of $op(i)$. A then tells him whether $op(i)$ is *compatible** with $|\Psi(d_i)\rangle$; if so, A and B know with certainty that, for that particular value of i, $d_i' = d_i$. All values of d_i and d_i' for which B has used a compatible measurement operator are kept and form the *tentative key;* other values are discarded.

   (b) As for BB84: the tentative key is processed further, in order to correct errors and to eliminate any information which may have "leaked" about the tentative key. This results in a *final key.*

* The meaning of "compatible" is defined as follows: if $d_i = 0$, then $op(i) = P_0$ is a compatible measurement operator; conversely, if $d_i = 1$, then $op(i) = P_1$ is compatible.

- *when A measures her particle and obtains "spin–up" (state $|\uparrow\rangle$), then subsequent measurement by B will produce "spin–down" (state $|\downarrow\rangle$);*
- *when B measures her particle and obtains "spin–down," then subsequent measurement by B will produce "spin–up."*

This assumes that A and B's measurements are compatible, as we shall explain shortly.

As the example demonstrates, although the first measurement always produces a random result, the second is completely deterministic. We will take up this issue again in Chapter 3, where we will see how this behaviour can be modelled using a Markov chain.

A certain complication arises with regard to the nature of the second measurement. The second measurement is completely deterministic *only* if it is *compatible* with the first. Physicists prefer the term "totally anticorrelated" to characterise compatible measurements in this context. But what exactly do these terms mean?

In protocol E91 for key distribution, the two users make measurements of particle spin using analysers whose principal axis is in one of three orientations; for user A, these are at angles $\phi_1^A = 0°$, $\phi_2^A = 45°$ and $\phi_3^A = 90°$ with respect to the vertical. For user B's analyser, possible orientations are at angles $\phi_1^B = 45°$, $\phi_2^B = 90°$, and $\phi_3^B = 180°$. When A measures particle 1 by setting her analyser's axis at a given angle ϕ, *so must B,* when measuring particle 2, in order that they obtain opposite results (e.g. $|\uparrow\rangle$ for A, and $|\downarrow\rangle$ for B).

So *the axes of the two analysers have to be parallel* in order to obtain totally anticorrelated results. There are only two cases in which A and B's results are guaranteed to be totally anticorrelated: when both analyser orientations are set at 45° or 90°.

NOTATION 2.3 *To quantify the possible measurement results for A and B, physicists use the so–called correlation coefficient $E(\vec{\alpha}_i, \vec{\beta}_j)$; this varies as a function of $\vec{\alpha}_i$, the orientation chosen by A to make the first measurement ($\vec{\alpha}_i$ has an angle ϕ_i^A with respect to the vertical), and of $\vec{\beta}_j$, the vector chosen by B to make the second measurement ($\vec{\beta}_j$ has an angle ϕ_j^B with respect to the vertical).*

The value of $E(\vec{\alpha}_i, \vec{\beta}_j)$ depends on the probabilities $P_{\pm\pm}(\vec{\alpha}_i, \vec{\beta}_j)$, which correspond to the outcomes of measurements with $\vec{\alpha}_i$ and $\vec{\beta}_j$. By way of example, the symbol

$$P_{+-}(\vec{\alpha}_i, \vec{\beta}_j)$$

denotes the probability of the following two events occurring together:

1. The first measurement, along direction $\vec{\alpha}_i$, produces "spin–up," and
2. The second measurement along direction $\vec{\beta}_j$, produces "spin–down."

NOTATION 2.4 *The subscripts $_+$ and $_-$ refer to "spin–up" (cf. value $+\frac{\hbar}{2}$), and "spin–down" (cf. value $-\frac{\hbar}{2}$), respectively.*

The definition of the correlation coefficient is thus:

$$E(\vec{\alpha}_i, \vec{\beta}_j) = P_{++}(\vec{\alpha}_i, \vec{\beta}_j) + P_{--}(\vec{\alpha}_i, \vec{\beta}_j) - P_{-+}(\vec{\alpha}_i, \vec{\beta}_j) - P_{+-}(\vec{\alpha}_i, \vec{\beta}_j) \tag{2.25}$$

Quantum theory requires that $E(\vec{\alpha}_i, \vec{\beta}_j) = -\left(\vec{\alpha}_i, \vec{\beta}_j\right) = \vec{\alpha}_i \cdot \vec{\beta}_j$. Importantly for our purposes, the total anticorrelation of A and B's results, which occurs when their analysers' axes are parallel, is expressed as follows:

$$E(\vec{\alpha}_2, \vec{\beta}_1) = E(\vec{\alpha}_3, \vec{\beta}_2) = -1 \tag{2.26}$$

But how does all this relate to key distribution? We have just seen that, if two users are supplied with a set of spin–entangled particles then, by measuring them, they may obtain totally anticorrelated results. Without exchanging any information directly, they can thus obtain sequences of bits which are the inverse of each other. Assuming all measurements are compatible, and no disturbance is produced prior to measurement, it is thus possible to establish a common sequence of bits. Two problems remain:

1. *Not all measurements are compatible.* The compatible choices of analyser orientations ($\vec{\alpha}_2, \vec{\beta}_1$ and $\vec{\alpha}_3, \vec{\beta}_2$) are only two possibilities out of nine in total. It is therefore highly likely that the second measurement will not be totally anticorrelated with the first.
2. *Eavesdropping may be possible.* What can be done to detect a potential eavesdropper?

The first problem is solved by performing *error correction,* as is done in BB84 and B92; non–compatible measurements generate errors with nonzero probability. Ekert[23] describes this step as follows:

> " After the transmission has taken place, A and B can announce in public the orientations of the analysers they have chosen for each particular measurement and divide the measurements into two separate groups: a first group for which they used different orientation of the analysers, and a second group for which they used the same orientation of the analysers. They discard all measurements in which either or both of them failed to register a particle at all. Subsequently, A and B can reveal publicly the results they obtained but within the first group of measurements only. "

The measurements in the "second group" are known to be totally anticorrelated after the above steps, and can be converted to a secret key.

Now, if no disturbance has occurred due to eavesdropping, the quantity

$$S = E(\vec{\alpha}_1, \vec{\beta}_3) + E(\vec{\alpha}_1, \vec{\beta}_2) + E(\vec{\alpha}_2, \vec{\beta}_3) - E(\vec{\alpha}_2, \vec{\beta}_2) \tag{2.27}$$

will be equal to $-2\sqrt{2}$, as required by the rules of quantum theory. When eavesdropping does occur, however, the value of S changes so that

$$-\sqrt{2} \leqslant S \leqslant \sqrt{2} \tag{2.28}$$

and this allows A and B to detect the eavesdropper. Evaluating this inequality is, effectively, a test for eavesdropping.

The full E91 protocol is listed in two parts, in Figures 2.4 and 2.5.

2.3.4 Comparing the Three Protocols

Quantum key distribution protocols rely on the two legitimate users' ability to obtain matching measurements on quantum particles. BB84 and B92 involve polarised photons, while E91 uses spin–entangled particles. The "matching measurements" are defined formally as follows.

1. In BB84, user B must choose the same basis for measuring the photons received as A used for preparing them to be certain of obtaining the correct result:

$$\mathsf{Meas}_{decb}\,|\Psi(d, encb)\rangle = d \;\text{ if }\; encb \equiv decb \tag{2.29}$$

2. In B92, user B must use a projection operator that is compatible with the state being measured:

$$\text{if } op(i)\{|\Psi(d_i)\rangle\} \neq \vec{0} \;\text{ then }\; d_i \;\text{ is recovered}$$

3. In E91, both users A and B must use parallel measurement vectors to achieve total anticor-

Figure 2.4 The E91 Protocol — Part 1.

1. Interaction with Source of EPR Pairs

(a) The EPR source generates a pair $\left(|\psi\rangle_{[1]}, |\psi'\rangle_{[2]}\right)$ in the joint state

$$|\psi_s\rangle_{[12]} = \frac{1}{\sqrt{2}}\left(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle\right)$$

The source transmits particle 1 to A, and particle 2 to B. The individual states $|\psi\rangle_{[1]}$ and $|\psi'\rangle_{[2]}$ are, of course, undefined prior to measurement.

(b) User A chooses an analyser orientation at random (one of $\vec{\alpha}_1, \vec{\alpha}_2, \vec{\alpha}_3$ at angles $\phi_1^A = 0°$, $\phi_2^A = 45°$ and $\phi_3^A = 90°$ respectively) and measures particle 1. The possible outcomes of the measurement are $|\psi\rangle_{[1]} = |\uparrow\rangle$ (or, in units of $\frac{\hbar}{2}$, $+1$) and $|\psi\rangle_{[1]} = |\downarrow\rangle$ (or, in units of $\frac{\hbar}{2}$, -1).

(c) User B chooses an analyser orientation at random (one of $\vec{\beta}_1, \vec{\beta}_2, \vec{\beta}_3$ at angles $\phi_1^B = 45°$, $\phi_2^B = 90°$ and $\phi_3^B = 180°$ respectively) and measures particle 2. The possible outcomes of the measurement are $|\psi'\rangle_{[2]} = |\uparrow\rangle$ (or, in units of $\frac{\hbar}{2}$, $+1$) and $|\psi'\rangle_{[2]} = |\downarrow\rangle$ (or, in units of $\frac{\hbar}{2}$, -1).

relation of results:

$$E(\vec{\alpha}_2, \vec{\beta}_1) = E(\vec{\alpha}_3, \vec{\beta}_2) = -1$$

We are ready to discuss the special processes of error correction *(reconciliation)* and privacy amplification.

2.3.5 Secret-Key Reconciliation and Privacy Amplification

Quantum key distribution protocols such as BB84, B92 and E91 necessarily involve a public discussion after quantum particles have been exchanged and measured. The purpose of such a discussion is to reconcile the bit sequences of the two legitimate users which may differ in certain positions due to channel errors and eavesdropping. A public discussion occurs in step 2(b) of BB84 and B92 (see Figures 2.2 and 2.3), and in steps 2(b) and 2(c) of E91 (see Figure 2.5) and is termed a *reconciliation protocol*.

Brassard and Salvail[24] define reconciliation as "the process of finding and correcting discrepancies between the secret key sent by Alice and the one received by Bob." They formalise the notion of a reconciliation protocol as an operation R^p which, when applied to bit sequences A and B, produces a sequence S as the two protocol users exchange some information Q through discussion over a public channel:

$$R^p(A, B) = [S, Q] \tag{2.30}$$

For an eavesdropper with access to the quantum channel linking the two users, reconciliation

Figure 2.5 The E91 Protocol — Part 2.

2. Public Discussion

(a) User B reports to user A his choice of analyser orientation. A replies by stating whether B's choice is compatible with A's choice. The only choices that are compatible are:

i. when A selects $\vec{\alpha}_2$, B selects $\vec{\beta}_1$ (since $\vec{\alpha}_2 \| \vec{\beta}_1$)

ii. when A selects $\vec{\alpha}_3$, B selects $\vec{\beta}_2$ (since $\vec{\alpha}_3 \| \vec{\beta}_2$)

(b) Steps 1(a) to 2(a) are repeated, i times. For all i, A's choices of analyser orientation form a sequence $\left\{\overrightarrow{A(i)}\right\}$. Similarly, B's choices are stored in a sequence $\left\{\overrightarrow{B(i)}\right\}$. We define COMMON as the set of positions i for which compatible measurements have been performed:

$$\text{COMMON} = \left\{ i \mid compat(\overrightarrow{A(i)}, \overrightarrow{B(i)}) = 1 \right\}$$

where the function $compat(\overrightarrow{A(i)}, \overrightarrow{B(i)})$ produces 1 if $\overrightarrow{A(i)}$ and $\overrightarrow{B(i)}$ are parallel:

$$compat(\overrightarrow{A(i)}, \overrightarrow{B(i)}) = \begin{cases} 1 \text{ if } \overrightarrow{A(i)} \parallel \overrightarrow{B(i)} \\ 0 \text{ otherwise} \end{cases}$$

The inverse of COMMON is $\overline{\text{COMMON}}$, the set of positions for which incompatible measurements have been performed:

$$\overline{\text{COMMON}} = \left\{ i \mid compat(\overrightarrow{A(i)}, \overrightarrow{B(i)}) = 0 \right\}$$

We write $\{R_A(i)\}$ for the sequence consisting of A's measurement results, and $\{R_B(i)\}$ for B's measurement results.

(c) User B reveals to user A, for all $i \in \overline{\text{COMMON}}$, the value of $R_B(i)$. A then compares this with $R_A(i)$; then the two users can compute the quantity S (see Equation (**2.27**)), which should be equal to $-2\sqrt{2}$ in the absence of an eavesdropper. This confirms that their results for $i \in$COMMON are totally anticorrelated and can be converted to a secret key.

is an opportunity to obtain information about their measurements. Thus a reconciliation protocol causes information to be *leaked* to the eavesdropper. This leaked information is $I_E(S|Q)$, the expected amount of Shannon information that an eavesdropper E can get about S given Q. Indeed, an *optimal* reconciliation protocol is designed so as to minimise $I_E(S|Q)$. Optimal protocols are impracticable[25], so in practice $I_E(S|Q)$ is never minimal. The purpose of *privacy amplification*[26], which must occur after reconciliation, is to eliminate all leaked information.

Typical primitives of reconciliation protocols include:

- *the application of hash functions,* drawn from a universal class[27], to the legitimate users' bit sequences;
- *interactive binary search,* for detecting discrepancies by exchanging $\lceil \log n \rceil$ bits at most (Brassard and Salvail call this, the primitive `BINARY`);
- *the probabilistic primitive* `CONFIRM`, which tests for equality of the two users' bit sequences;

- combinations of the above.

The protocols **Shell** and **Cascade,** also due to Brassard and Salvail (1994), use these primitives extensively; the former of the two is an "almost ideal" protocol (i.e. optimal and efficient), while the latter is highly efficient and preferred for implementations of quantum cryptography[28].

The standard reconciliation procedure common to these protocols is the *parity–check code*, which proceeds as follows:

1. The users' bit sequences are split into blocks of equal size; the block size is chosen so that each block contains no more than 2 errors.
2. The users compare the parity of each corresponding block in their bit sequences.
   (a) If the parity of block k of A's sequences matches the parity of the same block in B's sequence, they proceed to compute and compare parities for block $(k+1)$.
   (b) If the parities do not match, they recursively slice the block into two subsets, until they find the exact location of the error causing the mismatch. This is the so–called *interactive binary search*; as soon as the error is found, A tells B publicly how to correct the error.
3. The bit sequences of A and B are 'shuffled', or randomised, using an agreed common permutation. Then the process is repeated until all errors have been corrected. The outcome of reconciliation, a common error–free bit sequence k_{rec}, is termed the *reconciled key.*

In a nutshell, *privacy amplification*[29] consists of the following steps[30]:

1. A and B agree on the value of a constant integer τ.
2. A creates a random binary matrix K of dimensions $(N-\tau)\times N$, where N is the number of particles transmitted over the quantum channel. For example:

$$K = \begin{bmatrix} 0 & 1 & 1 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & \cdots & 0 \end{bmatrix}$$

3. A announces the matrix K to B over the public channel.
4. A and B apply K to the reconciled key in order to produce a shorter, final key k_F, of length $(N-\tau)$. The value of τ chosen in the first step must be chosen so that the amount of valid information the eavesdropper has about the final key is negligible.

According to Yamamoto (2004), the result of privacy amplification is to minimise the eavesdropper's mutual information about the key:

$$I_E(k_F; k_{rec}, \vec{v}) = H(k_F) - \underbrace{H(k_F|k_{rec}, \vec{v})}_{\text{negligible}} \tag{2.31}$$

In the above expression, the symbol $\vec{v}$ denotes all the classical data Eve has managed to intercept; it is known as the *eavesdropper's view* and will be revisited shortly.

2.3.6 Security Criteria for Quantum Key Distribution

In this section, we will be concerned with formalising exactly what it means to say that a quantum key distribution scheme is *secure.* We follow the conventions established in the work of Mayers (2001).

Remember that any scheme for key distribution must allow two or more users, who share no information initially, to eventually share a common, secret key. There are two crucial requirements for any such scheme, namely, an enemy should not be able to obtain the key and, whatever the enemy does, the user's key sequences should be identical. Key distribution protocols typically fail when the enemy has the power to *impersonate* each user, and when the enemy is able to *jam,* or block the channel joining both users. These problems can arise in quantum key distribution, but while the former of the two can be dealt with using *authentication* techniques[31], the latter cannot be solved by any protocol unless alternative channels are available.

In quantum key distribution (henceforth QKD), the legitimate users must check whether certain *validation constraints* are satisfied, in order to ensure that it is possible to establish a key at all; an example of a validation constraint is an upper bound on the tolerated number of errors on the quantum channel.

The quantities that arise in validation constraints and also in descriptions of security requirements for QKD are termed *security parameters.* The most significant security parameter is N, the total number of quantum systems (photons, EPR pairs) exchanged during a given QKD protocol. Other security parameters, including the tolerated error rate and the number of bits used to test for eavesdropping, are collectively denoted $\vec{\epsilon}$.

A fundamental security requirement for any QKD protocol is that the amount of information available to an enemy, which is some function $f(N, \vec{\epsilon})$, must decrease exponentially fast as N increases. Mayers' proof of the security of BB84 shows that this quantity is exponentially small in N for all $\vec{\epsilon}$, i.e. that

$$f(N, \vec{\epsilon}) \leqslant \underbrace{c(\vec{\epsilon})}_{\text{constant}} \cdot \exp(- \underbrace{g(\vec{\epsilon})}_{\text{constant}} \cdot N)$$

Security proofs usually involve fixed values of N and $\vec{\epsilon}$.

The requirement that a QKD protocol ensures *privacy* can be formalised as follows:

Definition 2.6 (Privacy Criterion) *We say that a quantum key distribution protocol is f–private (for some $f > 0$) if, for every strategy adopted by an eavesdropper,*

$$\sum_m \Pr\{m\} \cdot \Big(m - H_m(\vec{k} \,|\, \vec{v}) \Big) \leqslant f$$

where m is the length of the sender's key $\vec{k}$, and $\vec{v}$ is the eavesdropper's ***view*** *(consisting of all the classical data she received or generated during the protocol), and $H_m(\vec{k} \,|\, \vec{v})$, the conditional Shannon entropy of the*

key given the eavesdropper's view, is defined as follows:

$$H_m(\vec{k} \,|\, \vec{v}) = -\sum_{v} \sum_{k \in \{0,1\}^m} \Pr\{\vec{k}, \vec{v} \,|\, m\} \cdot \log_2 \Pr\{\vec{k} \,|\, \vec{v}\}$$

The privacy criterion requires that the key be uniformly distributed[32].

2.4 Dense Coding

There is a limitation inherent in the process of conjugate coding (section 1.2); the qubits that are transmitted have to be *distinguishable*, and this is only guaranteed when orthogonal states are used. Consequently, the sender can encode only one bit of information on a single qubit.

This limitation can be avoided using entangled states[33]. A scheme known as *dense coding* allows *two* classical bits of information to be encoded in one qubit. Roughly, it works as follows. In order to encode two bits of information, an entangled pair of particles is generated in one of the so–called *Bell states:*

$$\left|\Psi^-\right\rangle_{[\mathbf{12}]} = \frac{1}{\sqrt{2}} \left(|0\rangle_{[\mathbf{1}]} \; |1\rangle_{[\mathbf{2}]} - |1\rangle_{[\mathbf{1}]} \; |0\rangle_{[\mathbf{2}]} \right) \tag{2.32}$$

$$\left|\Psi^+\right\rangle_{[\mathbf{12}]} = \frac{1}{\sqrt{2}} \left(|0\rangle_{[\mathbf{1}]} \; |1\rangle_{[\mathbf{2}]} + |1\rangle_{[\mathbf{1}]} \; |0\rangle_{[\mathbf{2}]} \right) \tag{2.33}$$

$$\left|\Phi^-\right\rangle_{[\mathbf{12}]} = \frac{1}{\sqrt{2}} \left(|0\rangle_{[\mathbf{1}]} \; |0\rangle_{[\mathbf{2}]} - |1\rangle_{[\mathbf{1}]} \; |1\rangle_{[\mathbf{2}]} \right) \tag{2.34}$$

$$\left|\Phi^+\right\rangle_{[\mathbf{12}]} = \frac{1}{\sqrt{2}} \left(|0\rangle_{[\mathbf{1}]} \; |0\rangle_{[\mathbf{2}]} + |1\rangle_{[\mathbf{1}]} \; |1\rangle_{[\mathbf{2}]} \right) \tag{2.35}$$

There is a special class of unitary operations that can be performed on any one of these states *which will always produce another Bell state.* Subsequent measurement indicates precisely which operation was performed. According to Bouwmeester et al. (2000):

> "Identifying each combination [i.e. 00, 01, 10, and 11] with different information implies that we can encode two bits of information by manipulating both particles. [...] Identifying each Bell state with different information we can [...] encode two bits of information, yet, now by manipulating only one of the two particles."

So, it is possible to represent all possible combinations of two classical bit values using the four Bell states. The convention used in dense coding and quantum teleportation is hence known as the *Bell basis.*

Definition 2.7 (Lo (2003)) *Given a pair of qubits, a convenient basis to use is the* ***Bell basis,*** *which has Bell states as its basis vectors. The correspondence between the states of the classical bits and the Bell states,*

in this basis, is as follows:

$$\begin{aligned} |\Phi^+\rangle &\leftrightarrow 00 \\ |\Psi^+\rangle &\leftrightarrow 01 \\ |\Phi^-\rangle &\leftrightarrow 10 \\ |\Psi^-\rangle &\leftrightarrow 11 \end{aligned}$$

The transformations that can be performed on Bell states, which yield yet more Bell states are:

1. The identity operation (which leaves a state unchanged);
2. State exchange ($|0\rangle_{[2]} \longmapsto |1\rangle_{[2]}$ and $|1\rangle_{[2]} \longmapsto |0\rangle_{[2]}$, which transforms $|\Psi^+\rangle_{[12]}$ to $|\Phi^+\rangle_{[12]}$);
3. State–dependent phase shift (which transforms $|\Psi^+\rangle_{[12]}$ to $|\Psi^-\rangle_{[12]}$);
4. A combination of state exchange and phase shift (which takes $|\Psi^+\rangle_{[12]}$ to $|\Phi^-\rangle_{[12]}$).

An example of how the Bell states and the above operations can be used to transmit a pair of classical bits follows.

Example 2.12 *Bob wants to send the bit string 01 to Alice. They both use the convention:*

$$\begin{aligned} &\textit{a change from } |\Psi^-\rangle_{[12]} \textit{ to } |\Psi^-\rangle_{[12]} \textit{ represents 00} \\ &\textit{a change from } |\Psi^-\rangle_{[12]} \textit{ to } |\Phi^+\rangle_{[12]} \textit{ represents 10} \\ &\textit{a change from } |\Psi^-\rangle_{[12]} \textit{ to } |\Phi^-\rangle_{[12]} \textit{ represents 01} \\ &\textit{a change from } |\Psi^-\rangle_{[12]} \textit{ to } |\Psi^+\rangle_{[12]} \textit{ represents 11} \end{aligned}$$

Alice produces and sends state $|\Psi^-\rangle_{[12]}$ to Bob, who transforms it into $|\Phi^-\rangle_{[12]}$ by performing a combination of state exchange and phase shift. He sends this back to Alice; she interprets it as the bit string 01 after appropriate measurement.

2.5 Quantum Teleportation

When readers first encounter the term 'teleportation' in the literature on quantum information, they are very surprised; for, *teleportation* is usually a word encountered only in science fiction. However, quantum teleportation is a well–founded theoretical process which allows the transmission of a quantum state using only classical bits. It is, in a sense, the converse of dense coding.

Quantum teleportation allows a certain party, Alice, to send a qubit $|\Psi\rangle_{[1]} = \alpha\,|0\rangle_{[1]} + \beta\,|1\rangle_{[1]}$ to a colleague Bob, without delivering the particle directly to him. The trick is to use a pair of entangled particles (labelled 2 and 3 here); particle 2 is given to Alice, and particle 3 to Bob. The entangled state of these two particles is, say, the Bell state

$$|\Psi^-\rangle_{[23]} = \frac{1}{\sqrt{2}}\left(|0\rangle_{[2]}\ |1\rangle_{[3]} - |1\rangle_{[2]}\ |0\rangle_{[3]}\right)$$

Alice possesses particles 1 and 2, and Bob possesses only particle 3; whatever measurement Alice makes on particle 2, it will affect particle 3. There is a way of expressing the joint state of particles 1 and 2 in terms of Bell states, and when Alice measures both particles, she will project them onto a state directly related to the initial one, $|\Psi\rangle_{[1]}$. By telling Bob over a classical channel the result of her measurement, he can transform particle 3 and obtain the initial state.

2.6 Summary

In this chapter, we have laid the conceptual foundations for the observations and results of our work. We have discussed the principles and mathematics of quantum theory; the key ideas of information–theoretic security (pun intended); and the workings of the essential quantum protocols (for key distribution, dense coding and teleportation). Importantly, we have seen the importance of reconciliation and privacy amplification, tools which can be applied also outside the context of quantum protocols. The security criteria defined by Mayers have been described as well, and these will prove especially useful when, shortly, we will investigate the privacy of BB84 with automated verification tools.

3

Model Checking Techniques

The reader is by now in a position to appreciate that quantum protocols come in several varieties, and that the physical phenomena they employ allow for the accomplishment of important tasks in cryptography and general data communications. Clearly these protocols merit deeper investigation, with a view to proving their correctness and level of security, and to understand their many properties. This chapter discusses *model checking*, a well–established method in computer science for analysing the properties of a system and formally proving that it satisfies them.

The amount of intelligence a computing machine can demonstrate has always been hotly debated; however, computers nowadays do have limited ability in assisting human reasoning and constructing mathematical proofs automatically. This is the result of many years' development of formal, mechanical techniques for analysing system behaviour. Indeed, the study of *automated verification*, as it is known, is an important part of any computer scientist's training[1]. Automated verification techniques include *theorem proving* and *model checking*.

On the one hand, theorem proving tools provide mechanical assistance in developing logical proofs. To show the validity of a given statement, a *theorem prover* aids the user in applying the inference rules of a particular logic, and maintains a history of the steps taken.

Model checking, on the other hand, is a procedure involving three main steps: constructing an abstract model of a given system *(system specification)*; defining the properties desired of the system in a form that can be checked automatically *(property specification)*; and feeding the model into an appropriate software tool *(verification)*. A *model checker* then employs its built–in algorithms to prove, with little or no user intervention, whether the system model satisfies the properties given.

The latter of these two approaches to verification has been used in our work to investigate the properties of certain quantum protocols. In particular, we have used *logical* model checking[2] and *probabilistic* model checking[3] to assess the BB84 protocol (see section 2.3.1) for quantum key distribution.

Firstly, we will discuss, in turn, the three phases of model checking. This includes an enquiry into the syntax of description languages and temporal logic. We will then describe the operation of the specific tools, SPIN and PRISM. As PRISM handles probabilistic models, which are better suited for modelling quantum–mechanical behaviour, we spend some additional words

on relevant background to the tool. The chapter concludes with a detailed presentation of EPR–pair measurement, and how this can be specified formally using logic and Bayesian (conditional) probability.

3.1 System Specification, and Description Languages

System specification is arguably the most crucial step when performing model checking for a particular problem. The system to be analysed has to be described accurately in some general–purpose specification language; the description, or *model,* must incorporate all the salient features of the system's behaviour, and particularly those aspects of the system relevant to verification.

For a general communications protocol, there are several levels of abstraction at which a description can be made; frequently in protocol verification the emphasis is put on concurrency aspects and timing. It is of utmost importance that all users of a protocol interact in the correct order, and that data is not lost due to synchronisation errors. At this level of abstraction, however, it is immaterial what data representation is used, or whether a particular compression algorithm is involved. A suitable protocol model for analysing timing and other concurrency–related issues will abstract away from low–level considerations such as those just mentioned.

The situation is similar, but certainly more complex, in the analysis of *security* protocols. A suitable model in this case must take into account the details of encrypting and decrypting procedures, the availability and secrecy of keys, and the nature of the communication channels used. A specification language for security protocols will necessarily be more expressive than one intended for the protocols of the previous paragraph. A discussion of various specification formalisms and their features follows.

Ryan et al. (2001) use the process calculus CSP, originally developed by Hoare for describing concurrent processes[4], as a specification language for security protocols. CSP is a mathematical formalism that includes a rich set of operators with which one can express a great variety of behaviours. It is possible to use CSP in order to also define system properties. For security protocols, a given CSP description is typically rewritten in the *Casper* notation and then supplied to the FDR model checker[5].

In order to see how CSP can be used as a protocol specification language, consider the following trivial protocol:

1. User Bob creates a key k and sends it to user Alice.
2. Alice receives k and uses it to encrypt her message m; she sends the result to Bob.
3. Bob uses k for decryption and recovers the original message m.

In the style of CSP adopted in the previous reference[6], this protocol would be described using three process definitions, as shown in Figure 3.1.

The use of process calculi as specification languages for protocols is quite common. Robin Milner's CCS, which was developed on similar lines as CSP, evolved into the π–calculus[7] and is also well–suited for this task. Interestingly, Abadi and Gordon (1999) extended the π–calculus with cryptographic primitives and other features relevant to security protocols; the result is the so-called spi–calculus . In more recent work, Gay and Nagarajan used CCS to model the BB84

Figure 3.1 A description of a trivial protocol in the CSP style. The notation $c?k : Key$ denotes receiving a key k from channel c. Similarly $c!k$ indicates the transmission of k over channel c. The symbol $\{m\}_k$ stands for the cryptogram produced when encrypting message m under key k. Refer to Ryan et al. (2001) for details on syntax and more.

$$\begin{aligned} Alice &= (c?k : Key \rightarrow c!\{m\}_k \rightarrow Stop) \\ Bob(k) &= (c!k \rightarrow c?y : Text \rightarrow decrypt(y,k) \rightarrow Stop) \\ Protocol(k) &= (Alice \;||\; Bob(k)) \end{aligned}$$

quantum cryptographic protocol[8] and demonstrated the inability of an eavesdropper to succeed for a certain kind of attack. They subsequently developed a quantum process algebra, CQP, especially for the definition of quantum protocols[9]. CQP will be discussed in Chapter **5**.

Specification languages for model checking are by no means restricted to process calculi. The SPIN model checker, to be discussed in section 3.4, actually uses an imperative specification language known as PROMELA. We will now take a brief tour of PROMELA and the probabilistic variant PROBMELA.

PROMELA is an acronym for "protocol meta–language"[10] or, in its latest incarnation, "process meta–language"[11]. A model of a system in PROMELA includes *processes*, *channels*, and *state variables*. As in CSP, PROMELA uses the '**!**' operator for value transmission and '**?**' for value reception. There are control structures in PROMELA akin to the C programming language, including case selection, repetition and unconditional jumps. The main kinds of statements that can occur in a PROMELA program are[12]:

- assignments and conditions
- selections and repetitions
- *send* (!) and *receive* (?)
- *goto* and *break*
- *timeout*

A sample PROMELA program, taken from Holzmann (1991, p. 99), is given in Figure 3.1 to illustrate the syntax.

As a final example of a system description language, consider PROBMELA[13]. This language is built upon the syntax of PROMELA but is designed so that systems with probabilistic behaviour can be described. Whereas most description languages support only non–determinism in system models, in PROBMELA the *probabilities* of specific events can be supplied. For instance, the probabilistic if–statement:

PIF

```
[0.3]⇒ x:=2;
[0.3]⇒ x:=4;
[0.3]⇒ x:=6;
```

FIP

in PROBMELA performs one of the three assignment statements at random but with the respective probabilities specified in square brackets. This construct is particularly useful and applicable to

Figure 3.2 An example of an imaginary protocol which illustrates various features of the PROMELA language.

```
proctype A(chan q1)
{  chan q2;                    // declare channel name
q1?q2;                      // receive channel name q2 through the q1 channel
q2!123                      // send value 123 on channel q2
}
proctype B(chan qforb)
{  int x;                      // declare integer
qforb?x;                    // receive integer from qforb channel
printf("x=}
init
{
chan qname[2] = [1] of {chan}; // declare array of channels
chan qforb = [1] of {int}; // declare channel qforb
run A(qname[0]);            // run process A
run B(qforb);               // run process B
qname!qforb;                // send channel name qforb through the qname channel
}
```

the description of quantum protocols, which is the concern of this work. For instance, the classical outcome of measuring an ideal qubit can be expressed in PROBMELA thus:

PIF

```
[0.5]⇒ result:=0;
[0.5]⇒ result:=1;
```

FIP

We will have more to say about PROBMELA in Chapter 5, especially in connection with the *probUSM* model checker. The synthesis of many of the features of the languages presented in this section has given rise to QSPEC, a specification language designed by the author for describing quantum protocols[14]; QSPEC will be detailed in Chapter 5.

3.2 PROPERTY SPECIFICATION

Thus far, we have only considered means for describing system behaviour. However, this is insufficient for model checking, whose purpose is to demonstrate conclusively that a system operates in a desirable manner, and that it is free from design faults. Expressing precisely what features of a system are 'desirable' and exactly what constitutes a fault are the objectives of *property specification*. A *property* is any pattern of observable behaviour that a system should or should not exhibit; the function of a model checker is, thus, to show whether a system *satisfies* a given set of properties.

A property can be expressed as a *formula* in a given *logic*. Typically a system model is represented by a finite or infinite *automaton*, and the model checker must determine the truth or falsity of the statement

$$\sigma \models \Phi \tag{3.1}$$

which means[15], "the run σ of the automaton representing the system model *satisfies* formula Φ".

Properties of a system model are usually expressed as formulae in temporal logic; in some model checking systems however, the behaviour defined by a property is described explicitly instead. For example, the SPIN model checker converts properties written in LTL (Linear Temporal Logic) to patterns of behaviour ("never claims") expressed in PROMELA. The PRISM model checker requires that properties are expressed in PCTL. In the following sections, we proceed to describe logics for property specification which are, virtually without exception, founded on temporal logic.

3.2.1 Temporal Logic

Temporal logic, an important theme in philosophical study of logic, was first recommended by Pnueli (1977) as a tool for reasoning about program computations. While propositional logic allows one to make statements about Boolean variables using various connectives, temporal logic includes *modal operators* that quantify such statements over time.

If a program computation is regarded as a sequence of states

$$s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \quad \text{or, put otherwise,} \quad s_0 \xrightarrow{*} s_n$$

then one such state s_i may be regarded as the "present" moment in time, and all subsequent states are then moments in the future. The modal operators of temporal logic are used to quantify over present and future states.

Modal operators are applied to logical propositions; logical propositions consist of *atomic* propositions (which are regarded as uninterpreted symbols) and connectives such as $\neg$ ("not"), $\wedge$ ("and"), $\vee$ ("or") and $\Rightarrow$ ("implies"). An example of an atomic proposition is

$$a > 0$$

where a is a variable involved in some computation, and

$$(a > b) \wedge (b > 4) \Rightarrow (4 < b < a)$$

is an example of a logical proposition.

The most commonly used operators in temporal logic are $\Box$ ("henceforth"), $\Diamond$ ("eventually"), and $\circ$ ("next"). The formula $\Box P$ ("henceforth, P") means that P is true for all states in a computation. For a computation whose present state is s_i and whose "future" are all states s_j $(j > i)$, the formula $\Box P$ states that the proposition P is true in state s_i and will remain true for all s_j.

The formula $\Diamond P$ ("eventually, P") states that there is some point in the computation at which P is true. If s_i is the present state of a computation, then $\Diamond P$ means that, either P is true in s_i or it will be true at some time in the future.

Finally, $\circ P$ means that P is true in the *second* state of a computation (i.e. in state s_{i+1} if s_i is the present state). The three modal operators can be combined together to express more complex properties. Any unbroken sequence of operators is termed a *modality*, and the number of operators in the sequence is the *degree* of that modality.

Note that the "henceforth" and "eventually" operators are duals, so that $\neg\Diamond\neg P = \Box P$. Two commonly used combinations of these operators are,

$\Box\Diamond P$	which means "infinitely often, P", i.e. there is an infinite number of future states in which P is true. This is termed a *recurrence* property.
$\Diamond\Box P$	which means "eventually, henceforth P", i.e. there exists some state after which P remains true forever. This is called a *stability* property.

Therefore, the expression

$$(\Diamond\Box(a > b)) \wedge (\Diamond(b = 3))$$

is an example of a valid temporal formula that states, "eventually, a will remain greater than b forever, and there will come a point in the computation when $b = 3$." Note that the usual logical connectives ($\neg$, $\wedge$, $\vee$, $\Rightarrow$) are also applicable within temporal formulae.

3.2.2 Linear Temporal Logic (LTL) versus Computation Tree Logic (CTL)

There are two possible views regarding the nature of time, and each of these gives rise to a different class of temporal logic[16]. In mainstream model checkers, two kinds of temporal logic are actually used: *linear* and *branching* temporal logic[17]. This section is included for completeness, but does not contribute directly to the argument of the thesis; it may thus be skipped on a first reading, or at least until the reader is better acquainted with SPIN and PRISM.

Linear temporal logic (LTL) treats time in such a way that, each moment has a *unique* possible future. Thus any LTL formula is interpreted over linear sequence, and essentially describes the behaviour of a single program computation. LTL is used, for instance, in the model checker SPIN.

On the other hand, in a *branching* temporal logic (or *computation tree* logic, CTL), each moment in time may have several possible 'futures'. Therefore, formulae in such a logic are represented by infinite computation trees; each tree describes a possible computation of a non–deterministic program. The PRISM tool uses a branching temporal logic, known as PCTL (*probabilistic* computation tree logic). As will be discussed in section 3.5, PCTL caters for probabilistic computations, and the trees representing a given formula are labelled with probabilities.

Of interest is the observation that model–checking algorithms for branching temporal logics are significantly more efficient than those for linear temporal logics[18]. Given a transition system of size n, representing a system model, and a temporal formula of size m, model–checking algorithms for CTL run in time $\mathbf{O}(nm)$, while for LTL they run in time $n \cdot 2^{\mathbf{O}(m)}$. Despite this important difference in efficiency, both kinds of temporal logic are useful; there are formulae expressible in LTL that are not expressible in CTL and vice versa. Finally, it must be noted that some model checkers, such as *Cadence* SMV, are compatible with both kinds of temporal logic.

3.3 Verification

The final phase of a model checking solution to a given problem involves *applying an automated tool* to the system description and the specification of its properties. In the literature, properties that express desired system behaviour are termed *liveness* properties, while *safety* properties express the absence of undesirable system behaviour[19]. Clearly, the model checker is expected to prove that liveness and safety properties do hold for a given system.

A *logical* model checker, such as SPIN, produces a conclusive "yes" or "no" answer for verification, depending on whether a property holds or not. Such an answer is definitive and accurate only for the model of the system given to the tool; although useful conclusions can be drawn about the actual system under consideration, one must remember that a model often omits aspects of system behaviour for simplicity. Only when the model is shown to correspond precisely to the actual system, it is sound to trust the conclusions of a model checker in general.

A *probabilistic* model checker allows for the definition of random behaviour. Because probabilities can be included in system models, erratic or improbable behavior can be tagged as such, and the outcome of a verification is a given event's *probability of occurrence*. While logical model checking is concerned with what is *possible*, probabilistic model checking deals with the *likelihood* of several possibilities.

Enough said about model checking techniques in general; our attention now turns to the freely available SPIN[20] and PRISM[21] model checkers. Their workings will be discussed in detail with a view to qualifying them as useful reasoning and analysis tools for quantum protocols.

3.4 The SPIN Model Checker

In order to use SPIN to verify whether a PROMELA system description `model.pml` satisfies a LTL formula Φ, three steps must be performed:

1. SPIN must be invoked in a mode that recognises the LTL formula and generates PROMELA code (a so–called `never` claim) for its inverse:

   `spin -f '¬Φ'`

2. The output of the previous step must be appended manually to `model.pml`; then SPIN is re-invoked to generate an automatic verifier called PAN. PAN is a C program, particular to the model in question, which must be then compiled:

   `spin -a model.pml; gcc -o pan pan.c`

3. PAN is executed to determine whether Φ holds or not in the original model after all.

A `never` claim, which is what SPIN generates in step 1 above, is a pattern of behaviour that corresponds to formula Φ. SPIN's verification algorithms are incorporated into the the PAN program generated for the model; PAN tries to prove that such behaviour *never occurs* in the model. Therefore, a `never` claim is always a description of *undesirable* system behaviour, and PAN examines all the possible behaviours of a system to demonstrate that none are undesirable. If a counterexample to the claim is found, PAN shows it clearly in its output.

This aspect (`never` claims) of model checking in SPIN is quite non–intuitive and more involved than it could have been. One must always remember that, to prove that some LTL formula Φ is *satisfied* by a PROMELA model, SPIN must be supplied with its inverse, i.e. $\neg\Phi$. If a `never` claim for $\neg\Phi$ is shown to hold by PAN, it means that *there does not exist any state in the model under consideration in which* Φ *is not true; therefore* Φ *always holds.* To summarise:

REMARK 3.1 *In order to prove that* LTL *formula* Φ *is true for a given model, one must generate a* `never` *claim for* $\neg\Phi$*, i.e. invoke* SPIN *with the command* `spin -f '¬Φ'`.

SPIN uses LTL for property specification; we now briefly review LTL's syntax and semantics.

LTL includes the "always" (or "henceforth") operator $\Box$, the "eventually" operator $\Diamond$, the "next" operator $\circ$ as well as two more operators, "weak until" U and "strong until" $\mathscr{U}$. To explain precisely the meaning of these operators, it will be useful to borrow and extend some ideas and notation from section 3.2.1.

For a computation

$$s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \quad \text{or} \quad s_0 \stackrel{*}{\rightarrow} s_n$$

the symbol $s[i]$ will be used to denote the set of states from s_i to s_n. The notation $s \models \Phi$ means that formula Φ is satisfied in state s, while $s[i] \models \Phi$ has a similar meaning. In section 3.2 the notation $\sigma \models \Phi$ was also used, where $\sigma = \{s_0, s_1, \ldots, s_n\} = s[0]$.

Using these conventions, the expression

$$s[i] \models (p \,\mathsf{U}\, q) \tag{3.2}$$

holds:

- either when $s_i \models q$
- or when $s_i \models p$ and $s[i+1] \models (p \,\mathsf{U}\, q)$.

The meaning of the "strong until" operator is defined as follows; the expression

$$s[i] \models (p \,\mathscr{U}\, q) \tag{3.3}$$

holds when:

- $s[i] \models (p \,\mathsf{U}\, q)$ and, also,
- there exists some $j \geqslant i$ for which $s_j \models q$.

Defining the meaning of the other temporal operators is simpler. We have:

$$\begin{aligned} s &\models \Box p \;\text{ when }\; s \models (p \,\mathsf{U}\, \mathit{false}) \\ s &\models \Diamond q \;\text{ when }\; s \models (\mathit{true} \,\mathscr{U}\, q) \\ s[i] &\models \circ p \;\text{ when }\; s_{i+1} \models p \end{aligned}$$

The SPIN user's guide[22] contains a list of commonly used LTL formulae.

3.5 The PRISM Model Checker

Of the two verification systems described here, PRISM is in many ways more suitable for analysing quantum protocols. PRISM is an acronym for *probabilistic model checker,* and makes provisions for expressing probabilistic behaviour as a matter of principle. Whereas a logical model checker only states whether expression (**3.1**) is true or false, a tool such as PRISM computes the value of

$$P_{\sigma,\Phi} = \Pr\{\sigma \models \Phi\}$$

for given σ and Φ, that is to say, it computes the *probability* with which a particular formula is satisfied by a particular model. As will be explained in what follows, the models catered for by PRISM include specific values of probability for various behaviours and so do the formulas used in verification.

Probabilistic models and PRISM–like tools find applications in numerous areas of computer science where random behaviour is involved. Oft–cited applications are randomised algorithms, real–time systems and Monte Carlo simulation. However, the application of these ideas to quantum systems is, undoubtedly, an unrivalled success story for computer science. As explained in Chapter 2, the quantum phenomena of nature are inherently *random processes;* any reasoning about such phenomena necessarily has to account for this. It follows that PRISM is appropriate for analysing quantum protocols, and even quantum algorithms as well.

PRISM itself uses a built–in specification language that is based on Alur and Henzinger's REACTIVE MODULES formalism (see Kwiatkowska et al. (2004) for details). Using this language the user can describe probabilistic behaviour, either in the form of Markov decision processes (MDPS), or as discrete time Markov chains (DTMCS), or even as continuous time Markov chains (CTMCS). A few words are in order about these various kinds of *stochastic process.*

3.5.1 Distributions and Probabilistic Transition Systems

In sections 3.1 to 3.4, we have regarded computations as deterministic processes; in other words, we have treated computations simply as fixed sequences of states. In order to describe stochastic processes, a different computational model is needed. Rather than using a fixed, pre–determined set of state transitions, a random process attaches *probability distributions* to sets of states. For a countable set of states S, a probability distribution is defined as a function π that associates with each member s of S a specific probability. Formally,

$$\pi : S \mapsto [0,1] \text{ with } \sum_{s \in S} \pi(s) = 1 \tag{3.4}$$

The term *support* refers to the set of states whose associated probability under distribution π is non–zero[23]:

$$\text{Supp}(\pi) = \{s \in S : \pi(s) > 0\} \tag{3.5}$$

Another related concept is the *value of a distribution on a subset of states:*

$$\text{if } S' \subseteq S \text{ then } \pi(S') = \sum_{s \in S'} \pi(s) \tag{3.6}$$

NOTATION 3.1 *For a given set of states S, the symbol* $\text{Dist}(S)$ *denotes all possible distributions on S.*

EXAMPLE 3.1 *Consider a set of states $S = \{s_0, s_1, s_2\}$ and a distribution $\pi(S)$ such that:*

$$\begin{aligned} \pi(s_0) &= 0.3 \\ \pi(s_1) &= 0.45 \\ \pi(s_2) &= 0.25 \end{aligned}$$

Notice that $\sum_{s \in S} \pi(s) = \pi(s_0) + \pi(s_1) + \pi(s_2) = 1$. In this example there is no state with zero probability, so $\text{Supp}(\pi) = \{s_0, s_1, s_2\} = S$. If a subset $T \subset S$ is taken, say $T = \{s_0, s_2\}$, the value of π on T is

$$\pi(T) = \sum_{t \in T} \pi(t) = \pi(s_0) + \pi(s_2) = 0.55$$

A general means for describing a system with probabilistic behaviour is the *probabilistic transition system.* In such a model each step in a computation is represented by a *move,* or *transition,* from a particular state s to a distribution π of successor states. Whereas in a deterministic computation the successor of a given state s is known in advance, the successor is chosen here at random as specified by the probability distribution. The formal definition of a probabilistic transition system is as follows:

DEFINITION 3.1 (PROBABILISTIC TRANSITION SYSTEM) *A probabilistic transition system is a tuple*

$$\langle S, (\longrightarrow), \pi_{init} \rangle$$

where:

- *S is a non–empty finite set of states;*
- *$(\longrightarrow) \in S \times \text{Dist}(S)$ is a finite transition relation;*
- *$\pi_{init} \in \text{Dist}(S)$ is an initial distribution on S.*

NOTATION 3.2 *The notation $s \longrightarrow \pi$ is shorthand for $\langle s, \pi \rangle \in (\longrightarrow)$.*

EXAMPLE 3.2 *Suppose you need to describe the likelihood of various types of weather for tomorrow given today's weather. Assume that there are three possibilities for the weather on any day: s_0 (rainy), s_1 (windy), and s_2 (sunny). From empirical data, you have deduced the following:*

1. *If today it is raining, it is likely that it will be raining tomorrow too, but it is equally likely to be windy.*
2. *A windy day is likely to be followed by a rainy day with probability 0.7 or another windy day with probability 0.3.*

3. *A sunny day is rare, and is followed by a rainy day with probability 0.4, a windy day with probability 0.4 or another sunny day with probability 0.2.*

4. *To construct a model for the situation, it is fair to assume that all three kinds of weather are equally likely to start with.*

A probabilistic transition system (henceforth PTS*) provides a concise model of the above scenario. The elements of the* PTS $\langle S, (\longrightarrow), \pi_{init} \rangle$ *for this problem are:*

$$
\begin{aligned}
& S = \{s_0, s_1, s_2\} \\
& s_0 \longrightarrow \pi_0;\ s_1 \longrightarrow \pi_1;\ s_2 \longrightarrow \pi_2 \quad \textit{i.e.} \quad (\longrightarrow) = \{\langle s_0, \pi_0 \rangle, \langle s_1, \pi_1 \rangle, \langle s_2, \pi_2 \rangle\} \\
& \pi_{init}(s_0) = \pi_{init}(s_1) = \pi_{init}(s_2) = \frac{1}{3} \\
& \pi_0(s_0) = \pi_0(s_1) = 0.5,\ \pi_0(s_2) = 0 \\
& \pi_1(s_0) = 0.7,\ \pi_1(s_1) = 0.3,\ \pi_1(s_2) = 0 \\
& \pi_2(s_0) = 0.4,\ \pi_2(s_1) = 0.4,\ \pi_2(s_2) = 0.2
\end{aligned}
$$

3.5.2 Discrete Time Markov Chains

A probabilistic transition system is a convenient model for general stochastic processes. It is the preferred model in standard computer science discourse, due to its relevance for programming language semantics[24]. For historical reasons, it is worthwhile to mention the discrete time Markov chain as an alternative model for probabilistic behaviour. It is the basic kind of model used for modelling with PRISM.

Definition 3.2 (Discrete Time Markov Chain) *A discrete time Markov chain is a tuple* $\langle S, \mathbf{P} \rangle$ *where* S *is a finite set of states and* $\mathbf{P}$ *is a transition matrix. The transition matrix is defined as* $\mathbf{P} : S \times S \mapsto [0, 1]$ *with* $\sum_{t \in S} \mathbf{P}(s, t) = 1$ *for all* s*. A Markov chain satisfies what is known as the* ***Markov property****, namely that the current state,* s_n*, depends only on its predecessor,* s_{n-1}*. We write*[25]*:*

$$\Pr\{s_n = j \,|\, s_{n-1} = i,\ s_{n-2} = a,\ \ldots,\ s_0 = z\} = \Pr\{s_n = j \,|\, s_{n-1} = i\}$$

This differs from Definition 3.1 in that it uses a matrix instead of a relation to describe transitions, and that it imposes a dependency between adjacent states.

Example 3.3 *The scenario in Example 3.2 can also be described using a discrete time Markov chain with*

$$
\begin{aligned}
S &= \{s_0, s_1, s_2\} \\
\mathbf{P} &= \begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \end{array}
\begin{array}{c} \begin{array}{ccc} s_0 & s_1 & s_2 \end{array} \\ \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.7 & 0.3 & 0 \\ 0.4 & 0.4 & 0.2 \end{bmatrix} \end{array}
\end{aligned}
$$

3.5.3 MARKOV DECISION PROCESSES

Markov decision processes are of interest to computer scientists because, among other things, they can be used to specify nondeterministic scheduling of concurrent, probabilistic processes[26], or systems with both nondeterministic *and* probabilistic transitions. PRISM allows us to define and verify such systems of processes, and we mention them here merely for the sake of completeness.

DEFINITION 3.3 (BAIER ET AL. (2004)) *A Markov decision process (MDP) is a tuple*

$$\mathscr{M} = (S, \mathsf{Act}, (\longrightarrow), S_0)$$

*where S is a set of **states**, Act a finite set of **actions**, $(\longrightarrow) \subseteq S \times \mathsf{Act} \times \mathrm{Dist}(S)$ the **transition relation**, $S_0 \subseteq S$ the set of **initial states**. For $s \in S$, $a \in \mathsf{Act}$, we define $\mathrm{Steps}(s) = \{\langle a, \mu\rangle : s \xrightarrow{a} \mu\}$. State s is **terminal** if $\mathrm{Steps}(s) = \emptyset$. $\mathscr{M}$ is **finite–state** if S is finite, and **finitely branching** if $\mathrm{Steps}(s)$ is finite for all states s. A finite MDP is a MDP that is finite–state and finitely branching.*

3.5.4 PROBABILISTIC COMPUTATION TREE LOGIC (PCTL)

The probabilistic temporal logic PCTL is used in PRISM as the principal means for defining properties of systems modelled by discrete–time Markov chains or Markov decision processes. It is based on the branching temporal logic CTL[27] and allows us to define such properties as,

> "Predicate P holds within a certain amount of (discrete) time steps, with probability at least 0.4."

The syntax of PCTL is defined inductively by the following grammar[28]:

$$\Phi ::= \mathbf{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid [\phi]_{\bowtie p} \qquad \text{(state formulae)}$$
$$\phi ::= \Phi_1 \,\mathscr{U}^{\leqslant t}\, \Phi_2 \mid \Phi_1 \,\mathscr{U}\, \Phi_2 \mid \mathscr{X}\, \Phi \qquad \text{(path formulae)}$$

where $t \in \mathbb{N}$, $p \in [0,1] \subset \mathbb{R}$, $a \in AP$, $\bowtie \in \{>, \geqslant, \leqslant, <\}$. Other boolean operators such as $\vee, \Rightarrow$ also appear frequently in state formulae, but they can always be reduced to combinations of $\wedge$ and $\neg$. Note that the metavariable a ranges over AP, a set of *atomic propositions*.

The two kinds of formula express different types of system properties. State formulae consist either of time–invariant propositions or take the form $[\phi]_{\bowtie p}$; the meaning of this latter form is,

> " there exist, with probability $\bowtie p$ (i.e. greater than p, equal to p, or less than p), paths from the current state which satisfy ϕ. "

The intuitive meaning of path formulae, on the other hand, is as follows:

- the formula $\Phi_1 \,\mathscr{U}\, \Phi_2$ expresses the fact that Φ_1 holds continuously from the current state onward, *until eventually* Φ_2 becomes **true**. The operator $\mathscr{U}$ is known as "unbounded until".
- the formula $\Phi_1 \,\mathscr{U}^{\leqslant t}\, \Phi_2$ expresses the fact that Φ_1 holds continuously from the current state onward, *until* Φ_2 becomes **true** in at most t time units. The operator $\mathscr{U}^{\leqslant t}$ is known as "bounded until".

- the formula $\mathscr{X}\,\Phi$ expresses the fact that, in the next computational state, Φ will become **true**. The operator $\mathscr{X}$ is, hence, simply known as "next".

Using the "until" operators, we can express the temporal notions "*always, P*" and "*eventually, P*" discussed in previous sections. The reader is referred to the recent work of Ciesinski and Größer (see Bibliography) for more details.

3.6 Modelling the Measurement of an EPR Pair

We believe that, after a detailed exposition of various subtly related topics, there is nothing more satisfying than a solid application or example that fuses them together; this section presents precisely such an application.

Recall from section 2.1.3 the notion of entanglement, and specifically the example of the EPR state (**2.12**). Any entangled state is characterised by the fact that it cannot be expressed in terms of its component states. This means that, in an entangled pair, the individual state of each particle is unknown. Only the overall, *joint* state is known; when the particles are subjected to measurement, then their individual states "come into being" and entanglement is destroyed.

For a two–particle quantum system, there are exactly four states which correspond to entangled particles ("EPR pairs"); these are also referred to as *Bell states* (viz. section 2.4). The Bell states are defined in Equations (**2.32**) to (**2.35**), on page 38.

Measuring EPR pairs is distinctly different to measuring a general two–particle quantum system in that, after measuring one of the two particles, the outcome of measuring the other is fully known. The fact that, measuring one particle affects the outcome of measuring the other, independently of their physical separation, Einstein termed "spooky action at a distance"[29]. In the words of Rieffel and Polak (2000):

> " Measurement gives another way of thinking about entangled particles. Particles are not entangled if the measurement of one has no effect on the other. For instance, the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is entangled, since the probability that the first bit is measured to be $|0\rangle$ is $\frac{1}{2}$ if the second bit has not been measured. However, if the second bit had been measured, the probability that the first bit is measured as $|0\rangle$ is either 1 or 0, depending on whether the second bit was measured as $|0\rangle$ or $|1\rangle$ respectively. Thus the probable result of measuring the first bit is changed by a measurement of the second bit. On the other hand, the state $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$ is not entangled: since $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes (|0\rangle + |1\rangle)$ any measurement of the first bit will yield $|0\rangle$ regardless of whether the second bit was measured. Similarly, the second bit has a fifty-fifty chance of being measured as $|0\rangle$ regardless of whether the first bit was measured or not. Note that entanglement, in the sense that measurement of one particle has an effect on measurements of another particle, is equivalent to our previous definition of entangled states as states that cannot be written as a tensor product of individual states. "

3.6.1 Specifying EPR Pair Measurement Using Logic

How can the above scenario be described formally? It is possible to specify the problem in its full generality as a model using the language of mathematics, with a view to expressing all possible

outcomes exactly in temporal logic. The elements of the model are shown in Table 3.1.

NOTATION 3.3 (EPR STATES) *We will consistently use the symbols $\psi_1, \psi_2, \psi_3, \psi_4$ to represent the* EPR *states $|\Psi^-\rangle_{[12]}$, $|\Psi^+\rangle_{[12]}$, $|\Phi^-\rangle_{[12]}$, $|\Phi^+\rangle_{[12]}$ respectively.*

Table 3.1 Notation used to build a formal, abstract model of EPR Pair Measurement.

The EPR state ψ being modelled, one of:	$\psi_1 \equiv \|\Psi^-\rangle_{[12]}$, $\psi_2 \equiv \|\Psi^+\rangle_{[12]}$, $\psi_3 \equiv \|\Phi^-\rangle_{[12]}$, $\psi_4 \equiv \|\Phi^+\rangle_{[12]}$.
The particle (1 or 2) chosen to be measured first:	d
The current time instant:	i
The outcome of measuring the first chosen particle:	$m_i(d, \psi_j)$ (initial value $\perp$)
The outcome of measuring the other particle:	$m_i(2-d, \psi_j)$ (initial value $\perp$)
Computational state at time i:	$s_i(d, \psi_j)$

We proceed to construct an abstract model of EPR measurement using the conventions in the table. In particular, the computation $\sigma_{EPRm}(d, \psi_j)$ is a model of the problem in which two particles, labelled 1 and 2, are entangled in the EPR quantum state denoted by ψ_j, and particle d is chosen to be measured first. The computation involves three states, each denoting part of the measurement process[30]:

$$\sigma_{EPRm}(d, \psi_j) = s_0(d, \psi_j) \rightarrow s_1(d, \psi_j) \rightarrow s_2(d, \psi_j) \tag{3.7}$$

The state $s_0(d, \psi_j)$ is the *initial* state of the computation, in which the entangled pair ψ_j exists in isolation, and no measurement has yet been performed. The states $s_1(d, \psi_j)$ and $s_2(d, \psi_j)$ arise after the first and second measurements, respectively. Each computational state consists of the outcomes of measuring each of the two particles. We write

$$s_i = \Big(m_i(d, \psi_j);\ m_i(2-d, \psi_j)\Big)$$

where $m_i(d, \psi_j)$ is the outcome, at time instant i, of measuring particle d. In the initial state, no measurement has been performed, so the values of $m_0(d, \psi_j)$ and $m_i(2-d, \psi_j)$ are unknown and written as $\perp$.

EXAMPLE 3.4 *The notation used to describe the problem is heavy, so an example of its use is valuable. Consider the question: "How does this abstract model describe the outcomes of measurements on two particles which are jointly in the Bell state $|\Phi^+\rangle$ to start with? Assume that particle 2 is measured first."*

The scenario in the question is represented by the computation $\sigma_{EPRm}(2, \psi_4)$, since we are given that $d = 2$ and $\psi = \psi_4 \equiv |\Phi^+\rangle$. For the quantum state $|\Phi^+\rangle$ it is known that, possible outcomes for the first measurement are $|0\rangle$ and $|1\rangle$, or, in the formal notation,

$$m_1(2, \psi_4) = 0 \text{ or } 1 \text{ at random}$$

The second measurement depends on the first. We have in this case:

$$\begin{aligned} m_1(2,\psi_4) = 0 \quad &\Rightarrow \quad m_2(1,\psi_4) = 0 \\ m_1(2,\psi_4) = 1 \quad &\Rightarrow \quad m_2(1,\psi_4) = 1 \end{aligned}$$

which means that, if the first measurement (on particle 2) produces $|0\rangle$, then the second measurement (on particle 1) will too and vice versa.

The example illustrates an important aspect of the problem, namely that the computation is non–deterministic. There are always two possible outcomes for the first measurement, and thus the original definition of $\sigma_{EPRm}(d,\psi_j)$ (**3.7**), which is deterministic, is incorrect. Taking non–determinism into account, we rewrite $\sigma_{EPRm}(d,\psi_j)$ as

$$\sigma_{EPRm}(d,\psi_j) = s_0(d,\psi_j) \rightarrow \begin{cases} s_1(d,\psi_j) \\ s_1'(d,\psi_j) \end{cases} \rightarrow s_2(d,\psi_j) \tag{3.8}$$

For $a \in \{0,1\}$, we have

$$s_0(d,\psi) = \Big(m_0(d,\psi);\ m_0(2-d,\psi_j)\Big) = (\perp;\perp) \tag{3.9}$$

$$s_1(d,\psi) = \Big(m_1(d,\psi);\ m_1(2-d,\psi_j)\Big) = (a;\perp) \tag{3.10}$$

$$s_1'(d,\psi) = \Big(m_1'(d,\psi);\ m_1'(2-d,\psi_j)\Big) = (1-a;\perp) \tag{3.11}$$

$$\begin{aligned} s_2(d,\psi) &= \Big(m_2(d,\psi);\ m_2(2-d,\psi_j)\Big) \\ &= (a;a) \text{ or } (1-a;1-a) \text{ if } j=3 \text{ or } j=4 \text{ respectively, or} \qquad (3.12) \\ &= (a;1-a) \text{ or } (1-a;a) \text{ if } j=1 \text{ or } j=2 \text{ respectively.} \qquad (3.13) \end{aligned}$$

Proposition 3.1 *The equations (**3.8**) and (**3.9**) to (**3.13**) constitute a specification of* EPR *pair measurement in its full generality.*

Proof. Consider all distinct instances of the EPR pair measurement problem. There are four distinct quantum states $(\psi_1,\psi_2,\psi_3,\psi_4)$ and two cases to consider for each (i.e. when $d=1$ and when $d=2$). Using the axioms in (**3.9**) to (**3.13**), enumerate all eight possible values of $s_2(d,\psi)$ and compare them with the predictions of quantum theory for the first and second measurements. They will be found to correspond exactly. ■

To summarise: a formal, abstract model of EPR pair measurement has been presented. Using this model, various properties of the problem can be formalised:

1. **After the first measurement, the computation is deterministic.** For the EPR states $|\Phi^{\pm}\rangle$ (in the lighter notation, ψ_3 and ψ_4), the outcome of the second measurement is identical to the outcome of the first. Using notation from temporal logic we have:

$$\Diamond[m_2(2-d,\psi_3) = m_1(d,\psi_3)] \tag{3.14}$$

$$\Diamond[m_2(2-d,\psi_4) = m_1(d,\psi_4)] \tag{3.15}$$

i.e. "there will arise, *eventually*, a computational state in which the outcome of the second measurement is equal to that of the first."

For the EPR states $|\Psi^{\pm}\rangle$ (in the lighter notation, ψ_1 and ψ_2), the outcome of the second measurement is the complement of the outcome of the first. Using notation from temporal logic we have:

$$\Diamond[m_2(2-d,\psi_1) = 1 - m_1(d,\psi_1)] \tag{3.16}$$
$$\Diamond[m_2(2-d,\psi_2) = 1 - m_1(d,\psi_2)] \tag{3.17}$$

i.e. "there will arise, *eventually*, a computational state in which the outcome of the second measurement is complementary to that of the first."

2. **At the end of the computation, both particles have well–defined individual states.** Using notation from temporal logic, for $j \in \{1,2,3,4\}$ we have:

$$\Diamond\left[\Big(m_2(2-d,\psi_j) \in \{0,1\}\Big) \wedge \Big(m_2(d,\psi_j) \in \{0,1\}\Big)\right] \tag{3.18}$$

i.e. "there will arise, *eventually*, a computational state in which the outcomes of both measurements are known to be 0 or 1 exactly."

3. **The outcomes of the measurements are invariant with respect to the order in which the particles are measured.** That is to say, it does not make a difference which of the two particles is chosen to be measured first. So the value of $m_i(d,\psi_j)$ is independent of d:

$$\forall d \in \{1,2\} \quad : \quad m_1(d,\psi_j) = 0 \text{ or } 1 \text{ at random} \tag{3.19}$$
$$\forall d \in \{1,2\} \quad : \quad m_2(d,\psi_3) = m_2(d,\psi_4) = m_1(d,\psi_3) = m_1(d,\psi_4) \tag{3.20}$$
$$\forall d \in \{1,2\} \quad : \quad m_2(d,\psi_1) = m_2(d,\psi_2) = 1 - m_1(d,\psi_1) = 1 - m_1(d,\psi_1) \tag{3.21}$$

These properties can be proven manually, but this is far from an easy task. However a model–checker can prove them easily if supplied with the specification and the set of properties defined above.

Remember that for entangled particles, the order in which they are measured does not affect the outcomes. Therefore the use of d in the above model is actually redundant for EPR pairs. Still, the model is general enough to describe the effects of measuring *any* two–particle quantum system, where the order in which particles are measured is significant.

Note that the model described in this section is non–deterministic, and therefore the possible outcomes of the first measurement are equiprobable. It's high time to formulate the problem in terms of probabilities.

3.6.2 Specifying EPR Pair Measurement Using Probabilities

In any given expansion of a quantum state into component vectors, the scalar coefficients of each component vector are *probability amplitudes* (*cf.* page 17). The interpretation of these numbers as

square roots of probabilities is due to Max Born and is fundamental to relating the mathematical formalism of quantum mechanics to the results of observations[31]. In particular, for an entangled state such as (*cf.* page 20)

$$|\psi\rangle_{[\mathbf{12}]} = c_1 \cdot |u_1 v_1\rangle + c_2 \cdot |u_2 v_2\rangle = \frac{1}{\sqrt{2}} \cdot |u_1 v_1\rangle + \frac{1}{\sqrt{2}} \cdot |u_2 v_2\rangle$$

the probability of the whole state collapsing to $|u_1 v_1\rangle$ is $c_1^2 = \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$, while the probability of it collapsing to $|u_2 v_2\rangle$ is $c_2^2 = \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$. This collapse occurs as a consequence of the first measurement on any of the two component particles. In the first case, the result of the measurement is $|u_1\rangle$; in the second case the result is $|u_2\rangle$. This much can be inferred from the definition of an entangled state using the Dirac notation. We will now try to formalise these facts using the notion of conditional probability. The relevance of Bayesian probabilities in the study of "quantum behaviour" is discussed at length in the work of Caves, Fuchs and Schack[32].

The probability of a particular outcome in the second measurement is *conditional* on the outcome of the first measurement. Since this conditional probability is known for all the different EPR states, the outcome of the second measurement is always predictable. In order to write these probabilities, it is necessary to define appropriate events. EPR pair measurement is a random process with the following events, where ψ_j is one of the EPR states $\psi_1, \psi_2, \psi_3, \psi_4$:

$$\begin{array}{ll}
\mathsf{Meas}(\psi_j, 1, 0) & \text{Particle 1 of an EPR pair in state } \psi \text{ is measured as } |0\rangle\,. \\
\mathsf{Meas}(\psi_j, 1, 1) & \text{Particle 1 of an EPR pair in state } \psi \text{ is measured as } |1\rangle\,. \\
\mathsf{Meas}(\psi_j, 2, 0) & \text{Particle 2 of an EPR pair in state } \psi \text{ is measured as } |0\rangle\,. \\
\mathsf{Meas}(\psi_j, 2, 1) & \text{Particle 2 of an EPR pair in state } \psi \text{ is measured as } |1\rangle\,.
\end{array}$$

These events can be defined as predicates over the measurement outcomes $m_1(d, \psi_j)$ and $m_2(2-d, \psi_j)$, which were used in the previous section:

$$\begin{aligned}
\mathsf{Meas}(\psi_j, 1, 0) &= (m_1(d, \psi_j) = 0) \\
\mathsf{Meas}(\psi_j, 1, 1) &= (m_1(d, \psi_j) = 1) \\
\mathsf{Meas}(\psi_j, 2, 0) &= (m_2(2-d, \psi_j) = 0) \\
\mathsf{Meas}(\psi_j, 2, 1) &= (m_2(2-d, \psi_j) = 1)
\end{aligned}$$

The probabilities describing the outcomes of the first measurement are:

$$\Pr\{\mathsf{Meas}(\psi_j, 1, 0)\} = \Pr\{\mathsf{Meas}(\psi_j, 1, 1)\} = 0.5$$

That is to say, whatever the entangled state ψ_j, the outcomes of the first measurement are always $|0\rangle$ and $|1\rangle$, and both outcomes are equally probable. To express the possibilities that may arise in the second measurement, we use conditional events; but the outcomes depend on whether ψ_1 and ψ_2 are being modelled, or ψ_3 and ψ_4.

$$\begin{aligned}
\Pr\{\mathsf{Meas}(\psi_1, 2, 0) | \mathsf{Meas}(\psi_1, 1, 1)\} &= 1 \\
\Pr\{\mathsf{Meas}(\psi_2, 2, 0) | \mathsf{Meas}(\psi_2, 1, 1)\} &= 1 \\
\Pr\{\mathsf{Meas}(\psi_1, 2, 1) | \mathsf{Meas}(\psi_1, 1, 0)\} &= 1 \\
\Pr\{\mathsf{Meas}(\psi_2, 2, 1) | \mathsf{Meas}(\psi_2, 1, 0)\} &= 1
\end{aligned}$$

However:

$$\begin{aligned}
\Pr\{\mathsf{Meas}(\psi_3, 2, 0) | \mathsf{Meas}(\psi_3, 1, 0)\} &= 1 \\
\Pr\{\mathsf{Meas}(\psi_4, 2, 0) | \mathsf{Meas}(\psi_4, 1, 0)\} &= 1 \\
\Pr\{\mathsf{Meas}(\psi_3, 2, 1) | \mathsf{Meas}(\psi_3, 1, 1)\} &= 1 \\
\Pr\{\mathsf{Meas}(\psi_4, 2, 1) | \mathsf{Meas}(\psi_4, 1, 1)\} &= 1
\end{aligned}$$

All other conditional probabilities for the EPR pair measurement problem are identically zero.

Let's see how a probabilistic transition system can be used to describe the measurement of an EPR pair in one of the quantum states ψ_3 and ψ_4: here, the second measurement produces the same outcome as the first.

We construct a probabilistic transition system $\langle S, (\longrightarrow), \pi_{\text{init}} \rangle$ with five states:

s_0, the initial state, in which no measurement has been performed yet;
s_1, the state which arises when the first measurement produces $|0\rangle$;
s_2, the state which arises when the second measurement produces $|0\rangle$;
s_3, the state which arises when the first measurement produces $|1\rangle$; and
s_4, the state which arises when the second measurement produces $|1\rangle$.

The initial distribution π_{init} ensures that the only possible initial state is s_0; we have

$$\pi_{\text{init}}(s_0) = 1,\ \pi_{\text{init}}(s_k) = 0 \ \text{ for } \ k > 0$$

From the initial state, the next state represents the outcome of the first measurement. There is thus a transition from the initial state to a distribution of successor states π_0:

$$\begin{aligned}
&s_0 \rightarrow \pi_0 \ \text{ or } \ \langle s_0, \pi_0 \rangle \in (\longrightarrow) \ \text{ where:} \\
&\pi_0(s_1) = 0.5 \\
&\pi_0(s_3) = 0.5 \\
&\pi_0(s_k) = 0 \ \text{ for } k \notin \{1, 3\}
\end{aligned}$$

The above transition describes the fact that the first measurement can produce either $|0\rangle$ or $|1\rangle$ with equal probability.

From states s_1 and s_3 there is only a single transition in each case, describing the only possible

result for the second measurement:

$$
\begin{aligned}
&s_1 \longrightarrow \pi_1 \text{ or } \langle s_1, \pi_1 \rangle \in (\longrightarrow) \text{ where:} \\
&\pi_1(s_2) = 1 \\
&\pi_1(s_k) = 0 \text{ for } k \neq 2 \\
&s_3 \longrightarrow \pi_3 \text{ or } \langle s_3, \pi_3 \rangle \in (\longrightarrow) \text{ where:} \\
&\pi_3(s_4) = 1 \\
&\pi_3(s_k) = 0 \text{ for } k \neq 4
\end{aligned}
$$

So the set of all transitions is $(\longrightarrow) = \{\langle s_0, \pi_0 \rangle, \langle s_1, \pi_1 \rangle, \langle s_3, \pi_3 \rangle\}$. It is also possible to construct a simpler transition system with only three states (the initial state, a state for outcome $|0\rangle$, and a state for outcome $|1\rangle$) for this problem, but we will not discuss this further.

3.6.3 Model Checking EPR Pair Measurement with PRISM

In this section, we will present a trivial description of EPR pair measurement for the PRISM tool. The model consists of two parts: a definition of the EPR pair's behaviour when measured, and a definition of an observer's interaction with the particles. We discuss each definition in turn.

The first part of the model is listed and then explained below.

```
dtmc

module EPRpair
EPRstate : [1..4];

[init]       true -> 1/4 : (EPRstate'=1) + 1/4 : (EPRstate'=2) +
                     1/4 : (EPRstate'=3) + 1/4 : (EPRstate'=4);
[firstmeas]  true -> (EPRstate'=EPRstate);
[secondmeas] true -> (EPRstate'=EPRstate);
endmodule
```

All PRISM models commence with a declaration of the model type: a discrete–time Markov chain is indicated by the `dtmc` keyword, while `mdp` indicates a Markov decision process and `ctmc` a continuous–time Markov chain[33]. For our purposes, a `dtmc` will suffice. We proceed to define a module `EPRpair`, which initialises to one of the quantum states $\psi_1, \psi_2, \psi_3, \psi_4$ with equal probability $\frac{1}{4}$ (represented by consecutive values of a local integer variable, `EPRstate`). The quantum state of a physical EPR pair is, by definition, unknown and this is modelled by the fact that its value is revealed only after measurements, denoted `firstmeas` and `secondmeas`.

The labels `[init]`, `[firstmeas]` and `[secondmeas]` denote *synchronisations* in PRISM. Whenever they appear in more than one module, the corresponding actions are executed simultaneously. These labels appear also in the definition of the observer's behaviour, defined by the following module:

```
module Observer
comp_state : [0..2] init 0;    // computational state
outcome1 : [0..2];             // the value 2 represents an unknown state
```

```
outcome2 : [0..2];

[init]       (comp_state=0) -> (comp_state'=1);
[firstmeas]  (comp_state=1) -> 1/2 : (outcome1'=0) & (comp_state'=2) +
                               1/2 : (outcome1'=1) & (comp_state'=2);
[secondmeas] (comp_state=2) & ((EPRstate=1) | (EPRstate=2))
                            -> (outcome2'=1-outcome1) & (comp_state'=2);
[secondmeas] (comp_state=2) & ((EPRstate=3) | (EPRstate=4))
                            -> (outcome2'=outcome1) & (comp_state'=2);
endmodule
```

The details of the PRISM syntax will be described in section 4.1.1, but it is not essential for an understanding of the above code.

The outcomes of the observer's measurements are described here. The first measurement will always produce $|0\rangle$ or $|1\rangle$ with equal probability $\frac{1}{2}$. We know that, for ψ_1 and ψ_2, the outcome of the second measurement is the inverse of that of the first (hence `outcome2'=1-outcome1`). However, for the EPR states ψ_3 and ψ_4, the outcome of the second measurement is identical to that of the first (hence `outcome2'=outcome1` in this case).

What property can we check with PRISM for this model? We could ask for the probability that, if the outcome of the first measurement is $|1\rangle$, the outcome of the second will be $|1\rangle$. To express this we have to ask PRISM to compute:

```
P=? [((comp_state=2) & (outcome1=1)) => ((comp_state=2) & (outcome2=1))]
```

When commanded to compute this for the model described previously, PRISM produced the value **0.50**, which is indeed the probability that the second measurement matches the first when all possibilities are considered. When `EPRstate=1` or `EPRstate=2`, the property does not hold (since the second measurement of EPR pairs in states ψ_1 or ψ_2 produces the inverse of the first measurement). The reverse is true for `EPRstate=3` and `EPRstate=4`, so out of the four possibilities, exactly half satisfy the property.

3.7 Summary

In this chapter, we have discussed the technique of model checking in general, and the tools SPIN and PRISM in particular. We have looked at the syntax of several specification languages and also the characteristics of temporal logic. The distinction between linear temporal logic and branching temporal logic has been explained; and the measurement of an entangled pair of particles has been formally described using temporal logic and conditional probabilities. Finally, a PRISM model of EPR pair measurement has been detailed; this serves as a simple but effective example of PRISM's use.

4

Analysis of BB84 using PRISM and SPIN

Our work has been motivated by the observation by Nagarajan and Gay[1] that model checking is likely to be effective as a technique for analysing quantum protocols. Using the PRISM and SPIN model checkers, we have been able to investigate various properties of the quantum key distribution protocol BB84. This chapter presents the results of our efforts.

Quantum key distribution, despite an unconditional proof of security, has its limitations. The need for reconciliation highlights the fact that many errors are likely to occur during transmission, even in the absence of an eavesdropper. Furthermore, the process of reconciliation divulges valuable information to an eavesdropper, when she is present; the only way to resolve this is by processing the key further. Also, unless the users share some information in advance, the protocol fails to deliver a secret key in the presence of an eavesdropper. In this case, the only advantage of quantum key distribution over its classical counterparts is that it prevents the eavesdropper from passing unnoticed.

In Chapter 2 we skimmed over the fact that an eavesdropper has several alternative ways of attacking a QKD protocol. In Example 2.7, we assumed that the eavesdropper's chosen basis for measurement "replaces" the basis with which the photon currently in transit is polarised; this, however, is but one of several possibilities. We also need to define what is meant by the verb "replace" in this context.

We will consider two types of eavesdropping in detail in this chapter: *"intercept–resend"* and *"random–substitute"* eavesdropping. In general, the eavesdropping attacks possible on a QKD protocol are grouped into the following classes:

- **individual attacks,** in which the eavesdropper receives the particles, one at a time, and measures each one individually (thus combining an 'individual unitary interaction' with an 'individual measurement[2]);
- **collective attacks,** in which the eavesdropper applies a unitary transformation to each particle received, stores the result in a quantum memory, and measures all these results simultaneously (thus combining an 'individual unitary interaction' with a 'collective measurement');

- **coherent attacks,** in which the eavesdropper stores all the particles together in a quantum memory, and measures all of them simultaneously (thus combining a 'collective unitary interaction' with a 'collective measurement'). Coherent attacks are the most general type of attack possible.

Mayers' proof of the unconditional security of BB84 tackles the most general type of attack, i.e. coherent attacks.

Whichever attack the eavesdropper chooses to perform, she must remove each photon from the channel; it does not seem to be possible to directly measure a photon in transit. To measure the i-th transmitted photon, which is in state $|\Psi(d_i, b_i)\rangle$, the eavesdropper must choose a basis $\tilde{b}_i$. As we have seen in Chapter 2, to be certain of recovering the value of d_i, it must be that $\tilde{b}_i = b_i$; an incorrect choice of basis can only give the correct answer with probability 0.5. Moreover, in the latter case, the state is altered irreversibly.

Because the eavesdropper removes photons from the channel to make measurements, she has to replace each one of them. Not knowing with certainty what the transmitted quantum state of each photon was, the eavesdropper can only make a guess and prepare a substitute. In an intercept–resend scenario, she substitutes the i-th photon $|\Psi(d_i, b_i)\rangle$ with a new photon, in the state $|\Psi(\tilde{d}_i, \tilde{b}_i)\rangle$. In a random–substitute scenario, the eavesdropper chooses a basis $\hat{b}_i \in \{\boxplus, \boxtimes\}$ at random, and also a random data bit $\hat{d}_i \in \{0, 1\}$; she substitutes the i-th photon $|\Psi(d_i, b_i)\rangle$ with a new photon in state $|\Psi(\hat{d}_i, \hat{b}_i)\rangle$.

Which of the two attacks is most effective? How likely is it that an eavesdropper passes unnoticed? How likely is it that an eavesdropper makes consistently correct choices of basis for her measurements? These are questions we intend to address by performing model checking with PRISM. We will be using SPIN only in simulation mode.

Before going any further, we should formalise our intuition about the protocol by defining precisely the various events of interest. For the i-th photon transmitted, the event in which

$$(b'_i = b_i) \wedge (d'_i \neq d_i)$$

arises whenever an eavesdropper makes an incorrect choice of basis $\tilde{b}_i$ and is unlucky. This is a sufficient condition for detecting an eavesdropper, assuming a noiseless quantum channel. We will write P_{det} for the probability of detecting an eavesdropper when N photons are transmitted:

$$P_{det} = \Pr\{(b'_i = b_i) \wedge (d'_i \neq d_i)\} \text{ for any one value of } i < N$$

Later in the chapter we will show how PRISM may be used to see how P_{det} varies with N, for the two eavesdropping scenarios discussed above. Thus we will address the question regarding whether an eavesdropper can pass unnoticed, and the answer will be found to be mostly in the negative.

Now, out of N transmissions in total, there is a non–negligible probability $P_{1/2}$ that just over half will be intercepted and measured correctly by an eavesdropper. The eavesdropper makes a "correct" measurement of the i-th photon whenever

$$(\tilde{d}_i = d_i)$$

We write $E_c(i)$ for this event. Let $E_{c,m}$ denote the occurrence of $E_c(i)$ for m photons. Then, the probability that the eavesdropper obtains more than half the original bit values can be written:

$$P_{1/2} = \Pr\{E_{c,m}\} \text{ for } m > \frac{N}{2}$$

We will study the variation of this quantity, for different values of N, and for different eavesdropping scenarios.

How can the variation of these quantities be computed using PRISM? Remember from section 3.5 that PRISM's purpose is to determine $P_{\sigma,\Phi}$ for a given model σ (described in PRISM's input language) and Φ (expressed in PCTL). Actually, the latest version of the tool (at the time of writing, 2.0) allows the user to include parameters in models, and to compute $P_{\sigma,\Phi}$ for different values of these parameters. More formally, any PRISM model can be written in terms of several parameters $u_1, \ldots, u_k$:

$$\sigma = \sigma(u_1, \ldots, u_k)$$

Thus, PRISM can be used to compute $P_{\sigma(u_1,\ldots,u_k),\Phi} = \Pr\{\sigma(u_1, \ldots, u_k) \models \Phi\}$. Model checking $\sigma(u_1, \ldots, u_k)$ against Φ with PRISM, for specific values of $u_1, \ldots, u_k$, is termed an *experiment.*

In our models we have included two parameters:

N,	the number of photons transmitted by Alice onto the quantum channel;
$P_L = \Pr\{\tilde{b}_i = b_i\}$,	the probability that Eve is "lucky", by choosing the correct basis for her measurement. This parameter will remain constant at 0.5 for our purposes.

So we can ask PRISM to compute the values of

$$\Pr\{\sigma_1(N, P_L) \models \Phi\} \quad \text{and} \quad \Pr\{\sigma_2(N, P_L) \models \Phi\}$$

for different values of N and P_L (see below for an explanation of the notation). Φ is a suitable PCTL property that we wish to verify. We will discuss our choices of property Φ in section 4.2.

NOTATION 4.1 *We will use the symbol $\sigma_1(N, P_L)$ to denote the* PRISM *model of BB84 in section 4.1.1, which involves* ***intercept–resend*** *eavesdropping. N is the total number of photons transmitted by user Alice over the quantum channel, while P_L is the probability with which the eavesdropper obtains the correct measurement result despite an incorrect choice of basis.*

NOTATION 4.2 *We will use the symbol $\sigma_2(N, P_L)$ to denote the* PRISM *model of BB84 in section 4.1.2, which involves* ***random–substitute*** *eavesdropping.*

4.1 Two PRISM Models of BB84

What follows is a detailed walkthrough of the two PRISM models we have built for BB84. We describe the components of each model in turn.

4.1.1 MODEL OF BB84 WITH INTERCEPT-RESEND EAVESDROPPING

The source code for the first part of the model is shown below.

```
// Model type:
dtmc
// Number of bits to transmit:
const int N;
// Probability of obtaining correct value with wrong basis:
const double LUCKY;
// Definition of Quantum Channel:
module Channel
ch_state : [0..4];
ch_bas : [0..1];
ch_bit : [0..1];
[aliceput]     (ch_state=0) -> (ch_state'=1) & (ch_bas'=al_bas) & (ch_bit'=al_bit);
[evemeasure]   (ch_state=1) & (ch_bas=eve_bas)
                           -> (ch_state'=2);
[evemeasure]   (ch_state=1) & (ch_bas!=eve_bas)
                           ->   LUCKY   : (ch_state'=2) &(ch_bit'=ch_bit) +
                             (1-LUCKY) : (ch_state'=2) &(ch_bit'=1-ch_bit);
[eveget]       (ch_state=2) -> (ch_state'=3);
[eveput]       (ch_state=3) -> (ch_state'=4) & (ch_bas'=eve_bas) & (ch_bit'=eve_bit);
[bobget]       (ch_state=4) -> (ch_state'=0);
endmodule
```

The model of BB84 is a discrete–time Markov chain. We declare this by specifying the `dtmc` keyword. The parameters of the model are declared next (a specific value is assigned to these just prior to verification). The quantum channel is defined as a distinct entity, or agent, with which Alice, Bob and Eve interact; a PRISM module is always used to define an agent. The `Channel` module has three local variables, corresponding respectively to:

- the computational state for the channel;
- the basis with respect to which the photon currently on the channel is encoded (b_i ; the rectilinear basis is represented by value `0`, and the diagonal basis by value `1`); and
- the bit value (d_i) which is encoded by the photon currently on the channel.

Thus, when Alice sends a single photon in the quantum state $|\Psi(d_i, b_i)\rangle$, the channel's local variables `ch_bit` and `ch_bas` are updated to reflect, respectively, the values of d_i and b_i: the channel is capable of storing only one photon at a time. The lines of code following the declarations of `ch_state`, `ch_bas` and `ch_bit` describe what changes occur to these variables when Alice, Bob and Eve interact with it. When Alice is ready to send a newly encoded photon through the channel, she performs the `[aliceput]` action, which updates the variables in the channel module accordingly. When Eve attempts to measure the photon currently on the channel, she performs the `[evemeasure]` action, which results in the channel basis being altered to match Eve's decoding basis ($\tilde{b}_i$). Also, the bit value represented by the photon currently on the channel is altered to

match the result of Eve's measurement ($\tilde{d}_i$). This models the essence of intercept–resend eavesdropping. Before proceeding, a couple of words are in order about the syntax of commands in PRISM's input language.

A PRISM command takes the form

$$[a]\ g \longrightarrow u;$$

The meaning of this syntax is as follows: when action a is ready to be performed, the guard g, which is a boolean expression, is evaluated. If g is true, the right–hand side of the expression u, which consists of updates to variables, is executed. Frequently, it is convenient to introduce a variable representing computational state, which is included in the guard g. When such a variable is used, it serves as a program counter; thus in the sequence of commands

$$\begin{aligned}
&[a_1] \quad (comp_state = 1) \longrightarrow u_1 \,\&\, (comp_state' = 2);\\
&[a_2] \quad (comp_state = 2) \longrightarrow u_2 \,\&\, (comp_state' = 3);\\
&[a_3] \quad (comp_state = 3) \longrightarrow u_3 \,\&\, (comp_state' = 3);
\end{aligned}$$

the use of the *comp_state* variable ensures that the updates u_1, u_2, u_3 are executed in precisely that order. Note that the syntax of PRISM requires an update on the right–hand side of any command. Also note that updates of variables include the prime (′) symbol, in order to distinguish them from boolean expressions. Specific updates can be made to occur with a given probability; the syntax

$$[a]\ g \longrightarrow p_1 : u_1 + \cdots + p_k : u_k;$$

indicates to PRISM that, when performing action a, and guard g is true, only one of the updates u_i ($1 \leqslant i \leqslant k$) is performed; the probability of u_i being performed is given by p_i.

The definition of Alice's behaviour comes next:

```
module Alice
al_state : [0..5];
al_index : [1..N];
al_bas : [0..1];
al_bit : [0..1];
[]    (al_state=0) & (eve_state>0) -> 0.5 : (al_state'=1) & (al_bas'=0) +
                                      0.5 : (al_state'=1) & (al_bas'=1);
[]    (al_state=1) -> 0.5 : (al_state'=2) & (al_bit'=0) +
                      0.5 : (al_state'=2) & (al_bit'=1);
[aliceput]   (al_state=2) -> (al_state'=3);
[reveal]     (al_state=3) -> (al_state'=4);
[loop]       (al_state=4) & (al_index<N) -> (al_state'=0) & (al_index'=al_index+1);
[loop]       (al_state=4) & (al_index=N) -> (al_state'=5);
[stop]       (al_state=4) -> (al_state'=4);
[stop]       (al_state=5) -> (al_state'=5);
endmodule
```

As is the case for the module `Channel`, this module (`Alice`) has a computational state of its own, `al_state`, which changes at each step in the protocol. In BB84, Alice prepares N photons,

each encoding a bit value; from the point of view of PRISM, this can be modelled as the repetition, for N times, of the preparation of a single photon in a given state. This repetition is an iteration, or *loop*, with index variable `al_index`, ranging from 1 to N. The loop terminates when an eavesdropper is detected by Bob, as we will see shortly.

Alice's first action is to choose a basis `al_bas` (b_i) and also a bit `al_bit` (d_i) at random; each is chosen with equal probability (0.5). Then she prepares a photon in the quantum state $|\Psi(d_i, b_i)\rangle$ and performs the action `[aliceput]`, which places these values onto the channel. After this, she reveals her choice basis to Bob through action `[reveal]`. The `[loop]` and `[stop]` actions are used to indicate the repetition of this procedure, N times, to PRISM.

Bob's behaviour is described by the following module.

```
module Bob
bob_state : [0..7];
bob_bas : [0..1];
bob_bit : [0..1];
[]     (bob_state=0) & (eve_state>0) -> 0.5 : (bob_state'=2) & (bob_bas'=0) +
                                       0.5 : (bob_state'=2) & (bob_bas'=1);
[bobget]   (bob_state=2) & (bob_bas=ch_bas)
                         -> (bob_state'=3) & (bob_bit'=ch_bit);
[bobget]   (bob_state=2) & (bob_bas!=ch_bas)
                         -> 0.5 : (bob_bit'=ch_bit) & (bob_state'=3) +
                            0.5 : (bob_bit'=1-ch_bit) & (bob_state'=3);
[reveal]   (bob_state=3) & (bob_bas!=al_bas) -> (bob_state'=4);
[reveal]   (bob_state=3) & (bob_bas=al_bas) & (bob_bit=al_bit) -> (bob_state'=4);
[reveal]   (bob_state=3) & (bob_bas=al_bas) & (bob_bit!=al_bit) -> (bob_state'=7);
[loop]     (bob_state=4) & (al_index<N) -> (bob_state'=0);
[loop]     (bob_state=4) & (al_index=N) -> (bob_state'=6);
[stop]     (bob_state=6) -> (bob_state'=6); // no eavesdropping detected
[stop]     (bob_state=7) -> (bob_state'=7); // eavesdropping detected
endmodule
```

Firstly, Bob chooses a basis at random with respect to which to measure the photon received. Then, he performs a measurement; if his choice of basis matches the one Alice used, he recovers the bit value encoded therein correctly (`bob_bit'=ch_bit`). If his choice of decoding basis is incorrect, he obtains the correct value with probability 0.5. When Alice reveals to him her choice of basis, using action `[reveal]`, the computational state changes, depending on whether Eve is detected (`bob_state'=7`) or not (`bob_state'=4`). As soon as Eve is detected, the protocol is aborted (and the repetition, or *loop*, is broken).

Now, Eve also has to choose a basis at random for each of her measurements. If she is "lucky", although an incorrect basis is chosen, *she obtains the correct bit value.* This behaviour is described in the following module, which is otherwise similar to the one just described.

```
module Eve
eve_state : [0..4];
eve_bas : [0..1];
eve_bit : [0..1];
nc : [0..N];

[]            (eve_state=0) -> 0.5 : (eve_state'=1) & (eve_bas'=0) +
```

```
                              0.5 : (eve_state'=1) & (eve_bas'=1);
[evemeasure]  (eve_state=1) -> (eve_state'=2);
[eveget]      (eve_state=2) & (eve_bas=ch_bas)
                            -> (eve_state'=3) & (eve_bit'=ch_bit) & (nc'=nc+1);
[eveget]      (eve_state=2) & (eve_bas!=ch_bas)
                            -> LUCKY : (eve_state'=3) & (eve_bit'=ch_bit) & (nc'=nc+1) +
                           (1-LUCKY) : (eve_state'=3) & (eve_bit'=1-ch_bit);
[eveput]      (eve_state=3) -> (eve_state'=0);
endmodule
```

The variable `nc` counts the number of times that Eve's measurement result (whether she chooses the correct basis, or is "lucky") is correct, i.e. when $\tilde{d}_i = d_i$.

4.1.2 Model of BB84 with Random-Substitute Eavesdropping

We have seen that, in a random–substitute attack, Eve flips a coin to determine the values of $\hat{d}_i$ and $\hat{b}_i$, which she uses when preparing a substitute, in the state $\left|\Psi(\hat{d}_i, \hat{b}_i)\right\rangle$, for each photon she measures. This behaviour is reflected in an amended version of module `Channel`, shown below.

```
// Eve replaces 0 with probability REPLACE on channel and 1 with
// probability (1-REPLACE). She uses the same probabilities to
// replace the channel basis.
const double REPLACE = 0.5;
module Channel
ch_state : [0..6];
ch_bas : [0..1];
ch_bit : [0..1];
[aliceput]     (ch_state=0) -> (ch_state'=1) & (ch_bas'=al_bas) & (ch_bit'=al_bit);
[evemeasure]   (ch_state=1) & (ch_bas=eve_bas) -> (ch_state'=2);
[evemeasure]   (ch_state=1) & (ch_bas!=eve_bas)
                            -> LUCKY : (ch_state'=2) & (ch_bit'=ch_bit) +
                           (1-LUCKY) : (ch_state'=2) & (ch_bit'=1-ch_bit);
[eveget]       (ch_state=2) -> (ch_state'=3);
[eveput]       (ch_state=3) ->    REPLACE  : (ch_state'=4) & (ch_bit'=0)
                             + (1-REPLACE) : (ch_state'=4) & (ch_bit'=1);
[eveputbasis]  (ch_state=4) ->    REPLACE  : (ch_state'=5) & (ch_bas'=0)
                             + (1-REPLACE) : (ch_state'=5) & (ch_bas'=1);
[bobget]       (ch_state=5) -> (ch_state'=0);
endmodule
```

4.2 Desired Properties of BB84 and Verification Results

All quantum key distribution protocols must satisfy the following requirements:

1. The presence of an eavesdropper must be made manifest to the protocol users.

2. The protocol must ensure that the eavesdropper does not obtain the bits in the key ultimately shared by the two users.

Determining to what extent a particular protocol satisfies these requirements using model checking allows us to compute the probabilities P_{det} and $P_{1/2}$, and it is our objective to do this here for BB84. Specifically, defining a suitable PCTL formula for each requirement is all that is necessary to obtain these probabilities for the two PRISM models, $\sigma_1(N, P_L)$ and $\sigma_2(N, P_L)$. If Φ_1 is a formula corresponding to the event that an eavesdropper is detected, then

$$P^k_{det} = \Pr\{\sigma_k(N, P_L) \models \Phi_1\} \quad \text{with } k \in \{1, 2\} \tag{4.1}$$

is the probability of this event in model $\sigma_k(N, P_L)$.

Similarly, we can define a PCTL formula Φ_2 which holds whenever the eavesdropper is able to make more than $\frac{N}{2}$ measurements correctly. Thus we can write

$$P^k_{1/2} = \Pr\{\sigma_k(N, P_L) \models \Phi_2\} \tag{4.2}$$

for the probability of this particular event. Once we have defined the formulae Φ_1 and Φ_2, we will be in a position to conduct verification with PRISM and thence to compute the probabilities (**4.1**) and (**4.2**) for various values of N and P_L.

We have seen that, in both PRISM models of BB84, an eavesdropper is detected as soon as:

```
bob_state=7
```

It is, therefore, simple to define the formula for the first requirement:

$$\Phi_1 = \textbf{true}\ \mathscr{U}\ (bob\_state = 7) \tag{4.3}$$

In section 4.2.1, we will show how the probabilities

$$P^1_{det}(N, P_L) = \Pr\{\sigma_1(N, P_L) \models \Phi_1\} \tag{4.4}$$
$$P^2_{det}(N, P_L) = \Pr\{\sigma_2(N, P_L) \models \Phi_1\} \tag{4.5}$$

vary as N is increased P_L remains constant at 0.5.

With regards to verifying the second requirement: it is necessary to be able to count how many times on average the eavesdropper obtains a correct measurement result. The counter `nc` in the PRISM code for Eve's behaviour provides this information. Using this counter, we can formulate the event in which Eve obtains over half of Alice's bits correctly as follows:

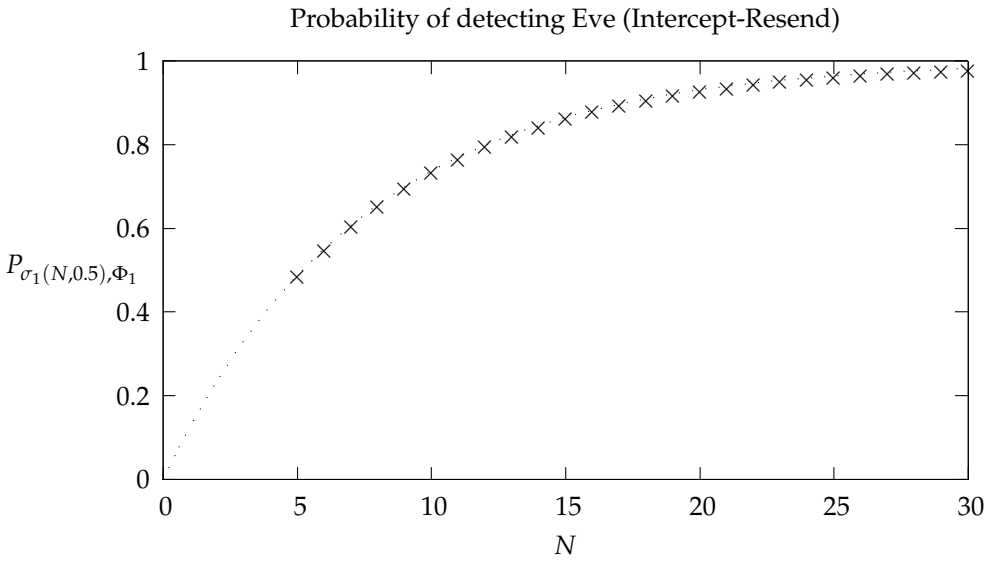
$$\Phi_2 = \textbf{true}\ \mathscr{U}\ \left(nc > \frac{N}{2}\right) \tag{4.6}$$

In section 4.2.2 we will study the variation of

$$\begin{aligned} & P^1_{1/2}(N, P_L) = \Pr\{\sigma_1(N, P_L) \models \Phi_2\} \\ \text{and} \quad & P^2_{1/2}(N, P_L) = \Pr\{\sigma_2(N, P_L) \models \Phi_2\} \end{aligned}$$

for different values of N, and constant $P_L = 0.5$.

Figure 4.1 The probability that Eve is detected in the BB84 Protocol while performing an intercept–resend attack, as a function of the security parameter N. The crosses indicate data points produced by PRISM, while the dotted curve is a non–linear least–squares fit to these points.



4.2.1 The Probability of Detecting an Eavesdropper

When PRISM is instructed to verify the property in (**4.3**) against $\sigma_1(N, P_L)$ and $\sigma_2(N, P_L)$, the tool performs standard procedure: it reads the model given, constructs an internal data structure, and then attempts to enumerate all the states in which Φ_1 holds. The version of Φ_1 which we have used for our analysis with PRISM is actually

$$\Phi_1' = \Phi_1 \{(ch\_state = 0) \wedge (al\_state = 0) \wedge (eve\_state = 0)\} \tag{4.7}$$

This formula (whose syntax is not part of pure PCTL, but is peculiar to PRISM) matches a subset of the states in which Φ_1 is true, namely those states in which the channel and users Alice and Eve are ready to proceed with the next transmission. So, in practice we used

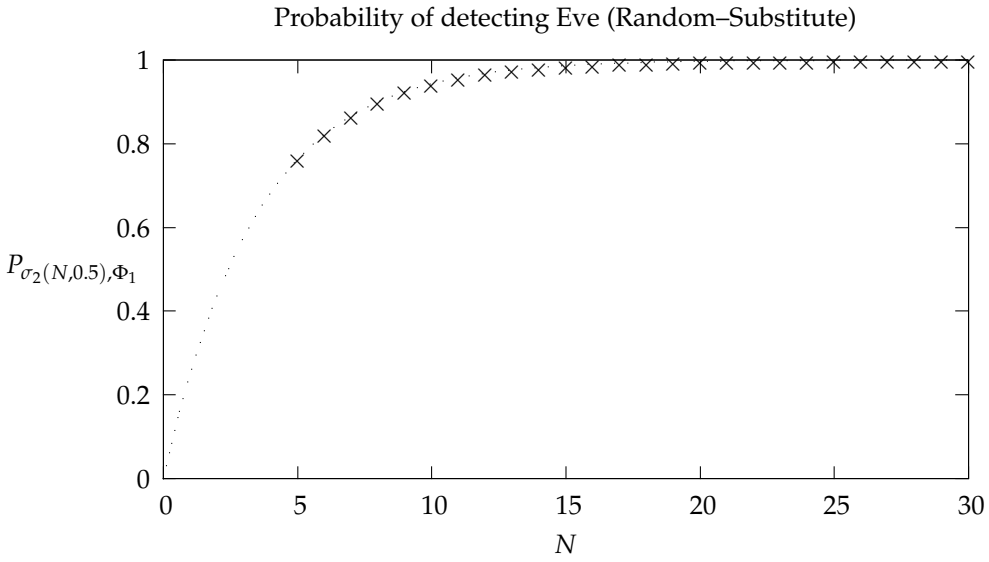
$$\Pr\{\sigma_k(N, P_L) \models \Phi_1'\} \tag{4.8}$$

as our definition of P^k_{det}, with $k \in \{1, 2\}$, instead of (**4.4**) and (**4.5**).

PRISM produces the results shown in Figures 4.1 and 4.2, when instructed to verify Φ_1' against $\sigma_1(N, 0.5)$ and $\sigma_2(N, 0.5)$ respectively, with $5 \leqslant N \leqslant 30$. Some representative results are given in the next table, to an accuracy of three decimal places.

N	$P^k_{det}(N, 0.5)$	$P^k_{det}(N, 0.5)$
5	0.487	0.763
15	0.865	0.987
25	0.965	0.999

Figure 4.2 The probability that Eve is detected in the BB84 Protocol while performing a random–substitute attack, as a function of the security parameter N.



It is clear from the two figures that, no matter whether Eve performs an intercept–resend or random–substitute attack, an increase in the number of photons transmitted leads to an exponentially greater probability of detecting her presence. The maximum value of N for which we have been able to compute $P_{\sigma_k(N,0.5),\Phi_1}$ is 30, and the value of P^1_{det} in this case is 0.9987, while P^2_{det} in this case is 0.9998.

Since the trend of the data is exponential, and exponential curves tend toward the horizontal axis asymptotically, there will never be —by the rules of calculus— a specific value of N for which P_{det} becomes unity. This means that there will always be a marginally small probability that the eavesdropper passes completely unnoticed. Mathematically, we can summarise these observations as follows:

$$\lim_{N\to\infty} P^k_{det}(N, 0.5) = \lim_{N\to\infty} P_{\sigma_k(N,0.5),\Phi_1} = 1$$

Experiments with values of P_L other than 0.5 produce the same result. The same figure has been reproduced in the text by Williams and Clearwater (1998).

Using the Marquardt–Levenberg nonlinear least squares algorithm for curve–fitting[3], we have been able to fit the values of $P_{\sigma_k(N,0.5),\Phi_1}$ produced by PRISM to a function of the form

$$f(N) = 1 - c_1 \cdot \exp[-c_2 \cdot N]$$

For the model $\sigma_1(N, 0.5)$, with intercept–resend eavesdropping, the values obtained for c_1 and c_2 are, respectively, 1 and 0.134. Thus, we can write that the probability of detecting an eavesdropper for this scenario is:

$$P^1_{det}(N, 0.5) \approx 1 - \exp[-(0.134)N] \tag{4.9}$$

For the model $\sigma_2(N, 0.5)$, with random–substitute eavesdropping, the values of $P^2_{det}(N, 0.5)$ are approximated as follows:

$$P^2_{det}(N, 0.5) \approx 1 - \exp[-(0.288)N] \tag{4.10}$$

All these results indicate that the BB84 protocol does ensure that, with probability arbitrarily close to unity, an eavesdropper's presence is made manifest to the legitimate users. We can safely conclude that requirement (1.) on page 67 is fulfilled by BB84.

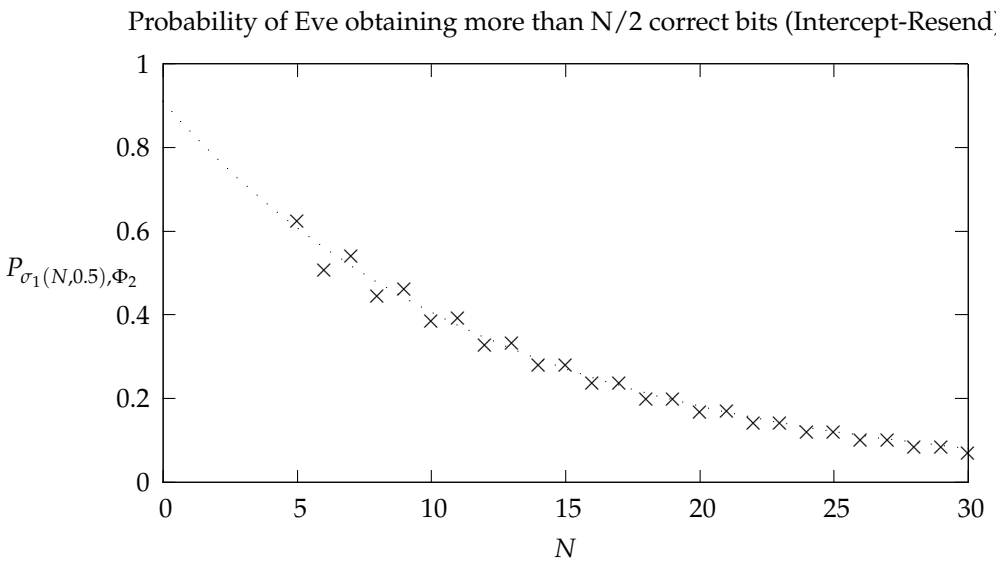
Finally, we can compare the variation of $P^1_{det}(N, 0.5)$ with that of $P^2_{det}(N, 0.5)$ to conclude as to which type of eavesdropping is more effective in masking Eve's presence. Since

$$P^1_{det}(N, 0.5) < P^2_{det}(N, 0.5) \tag{4.11}$$

it is clear that, of the two, intercept–resend is more effective.

4.2.2 The Number of Correct Bits Obtained by an Eavesdropper

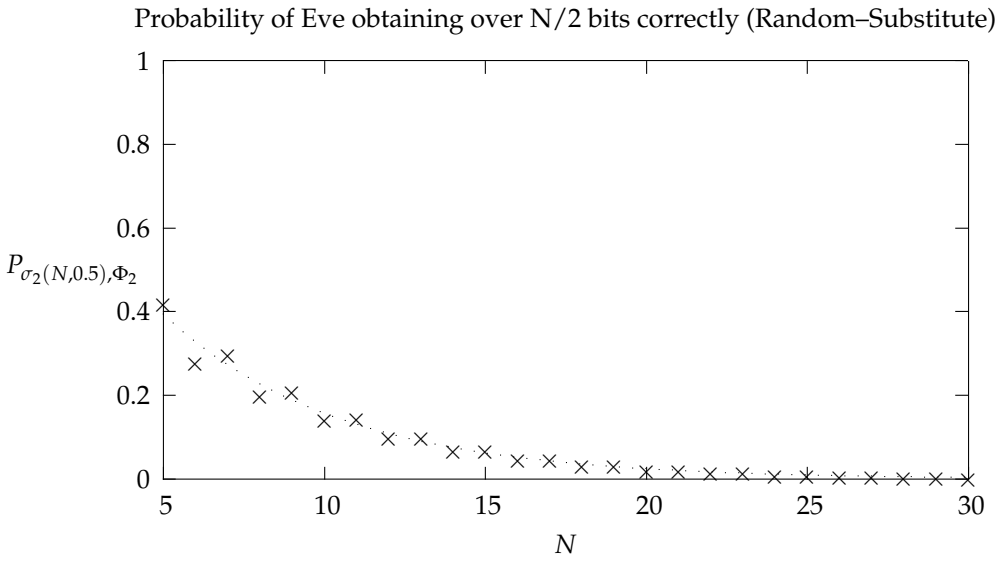
Figure 4.3 The probability that Eve, by performing an intercept–resend attack, obtains more than $\frac{1}{2}$ the total transmitted bits correctly in BB84, as a function of the security parameter N.



For the property expressed by the formula Φ_2, the results of model checking with PRISM are shown in Figures 4.3 and 4.4. It is evident from the graphs that the eavesdropper is increasingly less likely to obtain more than half Alice's original bit sequence correctly, as N is increased. In other words, the longer the sequence of photons transmitted by Alice, the less likely Eve is to measure half of it correctly.

Now, for both PRISM models, the variation of $P_{\sigma_k(N,0.5),\Phi_2}$ with N traces a decaying exponential

Figure 4.4 The probability that Eve, by performing an random–substitute attack, obtains more than $\frac{1}{2}$ the total transmitted bits correctly in BB84, as a function of the security parameter N.



curve; this corresponds to a function of the form

$$f(N) = c_1 \cdot \exp[-c_2 \cdot N]$$

We can fit the experimental results to this function as we did in the previous section; it suffices to note here that, for the intercept–resend scenario, the values of c_1 and c_2 produced by the fitting algorithm are 0.909 and 0.081 respectively. Thus we can write

$$P^1_{1/2}(N, 0.5) \approx (0.909)\exp[-(0.081)N]$$

Clearly, since we have exponentially decaying curves,

$$\lim_{N\to\infty} P^k_{1/2} = 0 \quad \text{for} \quad k \in \{1, 2\}$$

Note also that, in the security proof of Mayers (2001, p. 353), it is stated that:

> " In an information–theoretic setting, which is our case, a quantity f_N such as the amount of Shannon's information [sic] available to Eve must decrease exponentially fast as N increases. "

The results above corroborate this claim.

4.3 A SIMPLE SPIN SIMULATION MODEL OF BB84

The input language for the SPIN model checker, PROMELA, can actually be used effectively to describe BB84 and similar protocols; this is demonstrated in Appendix A, where such a description

is given. Although PROMELA was designed with a view to describing classical communication systems, it possesses two distinctive features that make it especially applicable to quantum protocols:

- non–determinism; and
- customisable data structures (records).

While PROMELA does not provide a way for specifying the probability of any particular event, its nondeterministic constructs are a practical tool for describing random behaviour. Thus, as discussed in the SPIN user guide[4], the following pattern of code represents a random choice between `N` possibilities:

```
if
:: true -> possibility1;
:: true -> possibility2;
         ...
:: true -> possibilityN;
fi
```

This type of construct necessarily treats all possibilities as if they were equiprobable.

The ability to define simple data structures in PROMELA makes it possible to define, for instance, a special type for qubits:

```
typedef qubit {
    real amplitude0;
    real amplitude1;
};
```

This code fragment models a qubit $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ by storing only the probability amplitudes α (`amplitude0`) and β (`amplitude1`). For BB84 we have used a similar type definition for polarised photon:

```
typedef particle {
    bit basis;
    bit value;
}
```

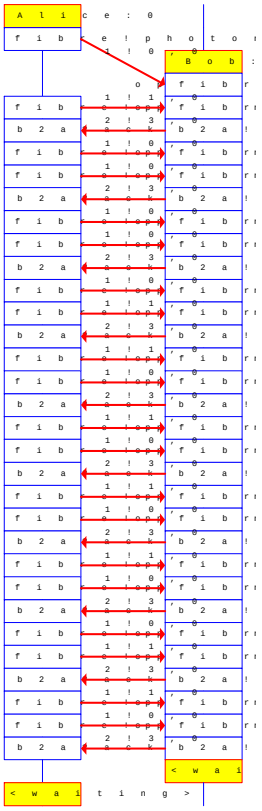
The component `basis` is 0 when a rectilinearly polarised photon is represented or 1 when a diagonally polarised one is represented.

Since *channels* for communication are declared explicitly in PROMELA, we model the quantum channel in BB84 simply as an optical fibre with a capacity of one photon:

```
chan opticalfibre = [1] of {particle};
```

The BB84 model in Appendix A also includes communications over classical channels (`a2b` and `b2a`) between Alice and Bob, which enable the two users to acknowledge to one another whether each photon has been transmitted and received successfully. In lieu of parity–check

Figure 4.5 A message sequence chart for BB84 produced by SPIN, assuming no eavesdropping.



reconciliation, it is assumed that Alice and Bob merely perform a public discussion in which measuring and preparation bases are compared for each photon, one at a time. The code for the various parts of the model is largely self–explanatory, and inline comments have been included to assist in its study.

DETAILS AND THE CRITERION FOR SECURITY AGAINST TAMPERING

Simulating the PROMELA model of BB84 using the command–line version of SPIN provides an elementary way of observing and understanding the workings of the protocol. The graphical version of the tool includes additional features for visualisation, amongst which the ability to create a message sequence diagram, such as the one in Figure 4.5.

If the process of reconciliation had been described fully in the PROMELA model, so that the correction of errors had been included explicitly, then Alice and Bob's keys could be shown to match exactly. The requirement that, at the end of the protocol, the two users share a key is known as the criterion for *security against tampering*. The criterion is captured by the following

LTL formula:

$$\forall i : \Diamond(\texttt{measurements}[\texttt{i}] = \texttt{initialkey}[\texttt{i}]) \tag{4.12}$$

("eventually, the bit sequence produced by Bob's measurements and subsequently subjected to reconciliation, will match exactly the bit sequence chosen by Alice prior to transmission.")

A property such as (**4.12**) could be verified using SPIN against a fuller model of BB84; that would constitute a proof that BB84 does indeed provide security against tampering, and would be in agreement with the information–theoretic proofs currently available in the literature[5].

4.4 SUMMARY

In this chapter, we have discussed the strengths and limitations of quantum key distribution with BB84, as well as the possible attacks an eavesdropper could attempt. Two PRISM models of BB84 were presented in detail, one for an intercept–resend attack and one for a "random–substitute" attack. By verifying two PCTL properties against these models with PRISM, we have been able to establish that:

- the probability that an eavesdropper is detected by the legitimate users increases exponentially with the security parameter N;
- the probability that an eavesdropper obtains a correct measurement result for over half the transmissions, decreases exponentially with N;
- of the two eavesdropping attacks, intercept–resend is more likely to mask an enemy's presence.

Also, we have seen that it is possible to use logical model checkers, such as SPIN, to prove whether BB84 provides security against tampering. Using SPIN merely as a simulation tool, as we have done, seems to be a useful technique for understanding and visualising steps in the protocol.

5

Specification Formalisms and Related Work

In chapter 4, we saw how the method of automated verification can be fruitfully applied to the analysis of quantum protocols. Model checking has been used many times before for protocol design and testing in a classical (i.e. "non–quantum") setting[1]. Alternative approaches to this task are possible and numerous: while the technique of model checking is very valuable, the use of process algebra and formal logic are equally useful for modelling protocols. This chapter aims to discuss these latter techniques and how they may be applied to the task of specifying *quantum* protocols. We will focus on the latest developments in the field.

The use of spin as a vehicle for investigating BB84 has highlighted the importance and relevance of logical methods. We have, till now, paid attention only to temporal logic, since it is the *lingua franca* for property specification in model checking. There exist, however, other relevant kinds of modal logic, such as the logic of *knowledge* and the logic of *belief*. The ban (Burrows–Abadi–Needham) logic, which is frequently used for the analysis of security protocols, is a logic of belief. Using the constructs of ban logic, we can formalise our intuition regarding "who knows what" in a given protocol. Interestingly, Ron van der Meyden has even developed a model checker[2] for the logic of knowledge.

Recent work by Meyden and Patra (2003) has included a logic of knowledge and time for quantum systems, which has been used to specify some of the properties of the B92 and quantum teleportation protocols. We will summarise the syntax and semantics of this "quantum logic" in section 5.3. Prior to discussing logical methods, we will direct our attention toward the quantum process algebras, cqp *(Communicating Quantum Processes)* and *QPAlg*, which were presented lately at the qpl 2004 workshop in Turku, Finland.

How are process algebras relevant to protocol analysis? Firstly, we are reminded that a process algebra provides a high–level, theoretical abstraction of a computational process. The development of process algebras csp (by Tony Hoare[3]) and ccs (by Robin Milner[4]) in the sixties and seventies supplied solid mathematical foundations for the theory of concurrency. Since communication protocols, by definition, describe concurrent processes, they have always been ideal targets for specification using process algebra. The most oft–cited classical protocol in the process algebra literature is the alternating–bit protocol[5], but this does not concern us here. Of interest to us is the fact that process algebras often include *process equivalences*, which can be used to de-

velop manual proofs that a protocol meets its specification. At the time of writing, neither CQP nor *QPAlg* possess such equivalences. We will discuss CQP in section 5.1, and *QPAlg* in section 5.2. Section 5.5 discusses the possibilities offered by quantum system simulators, such as QCSim[6].

In section 5.4, we will discuss the design of an imperative specification language for quantum protocols, named QSpec, which the author presented earlier this year at the PREP2004 conference in Hertfordshire, England[7].

This chapter also presents, if somewhat briefly, alternatives to the PRISM model checker and its input language. The research group of Prof. Christel Baier (in Bonn, Germany) have lately proposed a probabilistic specification language, PROBMELA, which is cousin to the specification language for SPIN. They have also developed a model checker, *probUSM*, that uses PROBMELA for system models. We will discuss this further in section 5.6.

5.1 The CQP Formalism

The CQP formalism[8] is based on the π–calculus[9], to which it adds various primitive operations for manipulating qubits. Its distinctive characteristics are:

- an expression language which allows for direct measurement and transformation of quantum state;
- the ability to define processes involving a combination of quantum and classical operations;
- the ability to explicitly declare and manipulate quantum communication channels;
- a static type system which ensures type preservation for all processes, and unique ownership of each qubit in a system.

Figure 5.1 shows a simple model of the quantum teleportation protocol, expressed using CQP. The model consists of three *agents* (*Alice, Bob, System*) who interact with the ultimate goal of teleporting an entangled pair of particles (x, y) using a classical channel c. Each agent takes a certain number of arguments, which are declared explicitly. The syntax $s : \hat{}[\mathsf{Qbit}]$, for instance, declares a channel s capable of storing qubits. Each computational step in the protocol is indicated by an *action*, such as $\{x* = \mathsf{H}\}$ or $s![x]$, and actions are delimited by the prefix symbol (.), following the conventions of CCS and the π–calculus.

Figure 5.1 A CQP model of the quantum teleportation protocol.

$$
\begin{aligned}
\mathit{Alice}(x : \mathsf{Qbit},\ c : \hat{}[0..3], z : \mathsf{Qbit}) &= \{z, x* = \mathsf{CNot}\}\ .\ \{z* = \mathsf{H}\}\ .\ c![\mathsf{measure}\ z, x]\ .\ \mathbf{0} \\
\mathit{Bob}(y : \mathsf{Qbit},\ c : \hat{}[0..3]) &= c?[r : 0..3]\ .\ \{y* = \sigma_r\}\ .\ \mathit{Use}(y) \\
\mathit{System}(x : \mathsf{Qbit},\ y : \mathsf{Qbit},\ z : \mathsf{Qbit}) &= (\mathsf{new}\ c : \hat{}[0..3])\ \ (\mathit{Alice}(x, c, z) \mid \mathit{Bob}(y, c))
\end{aligned}
$$

To simulate the execution of the teleportation protocol, all that is necessary is to supply a qubit z to agent *System* and to evaluate the result. The interested reader is referred to the original paper[10] for details, which includes models of quantum coin flipping and quantum bit commitment in addition to teleportation.

Formal Syntax of CQP, the Type System and its Significance

According to the inventors of CQP[11], one of the most crucial design considerations for this formalism has been the inclusion of types. The original CQP paper goes to great lengths to describe the type system and its implications. The types available for use in CQP models are defined by the following grammar:

$$T ::= \mathsf{Int} \mid \mathsf{Unit} \mid \mathsf{Qbit} \mid \hat{}[\tilde{T}] \mid \mathsf{Op}(1) \mid \mathsf{Op}(2) \mid \cdots \mid \mathsf{Op}(n)$$

The first three types are self–explanatory (note that a Unit type is commonly associated with side–effects in functional programming). The form $\hat{}[\tilde{T}]$ is the type of a communication channel capable of storing n–tuples with type $\tilde{T} = T_1, \ldots, T_n$. The forms $\mathsf{Op}(n)$ denote the types of operators on n qubits.

The values, expressions and processes in a CQP specification involve the following syntax, respectively:

$$\begin{aligned} v &::= \mathsf{a} \mid \mathsf{b} \mid \cdots \mid \mathsf{z} \mid 0 \mid 1 \mid \cdots \mid 9 \mid \mathsf{unit} \mid \mathsf{H} \mid \cdots \\ e &::= v \mid \mathsf{measure}\ \tilde{e} \mid \tilde{e}* = e \mid e + e' \\ P &::= \mathbf{0} \mid (P||P) \mid e?[\tilde{x} : \tilde{T}].P \mid e![\tilde{e}].P \mid \{e\}.P \mid (\mathsf{new}\ x : T)P \mid (\mathsf{qbit}\ x)P \end{aligned}$$

Values, v, consist of variables, literals and unitary operators such as H (the Hadamard transformation, see Gruska (1999) for an explanation of H). Expressions, e, can be values, measurements, applications of unitary operators (the notation $\tilde{e}* = e$ stands for the application of expression e to the tuple of expressions $\tilde{e} = e_1, \ldots, e_n$), or applications of data operators (as in $e + e'$). The syntax of processes is similar to that of the π–calculus, with a null process $\mathbf{0}$, parallel composition, value reception and transmission, actions and declarations. The form $(\mathsf{new}\ x : T)P$ declares a new channel x for use by process P, while $(\mathsf{qbit}\ x)P$ declares a qubit variable x local to P.

The semantics of expressions and processes in CQP are defined, respectively, using reduction rules between configurations of the form

$$(\sigma; \phi; e) \text{ and}$$
$$(\sigma; \phi; P)$$

where σ stores all declared qubits, ϕ is a list of channel names, e is a given expression and P a process. The reduction rules describe probabilistic transitions between such configurations.

By applying the typing rules for CQP it is possible to eliminate design errors in a particular specification; any expression which is not well–typed signposts a logical error. A *type–checker* can be built on these rules so as to detect such errors automatically. Using *typing judgements* of the form

$$\begin{aligned} \Gamma; \Sigma; \Phi &\vdash e : T \\ \text{and } \Gamma; \Sigma; \Phi &\vdash P \end{aligned}$$

Gay and Nagarajan have been able to prove a type–preservation theorem, and also that each qubit in a particular CQP model is guaranteed to belong to only one process at a particular time; the

exchange of a qubit between two processes thus constitutes a physical "transfer of ownership."[12]

5.2 QPAlg: Another Quantum Process Algebra

Concurrently with the CQP formalism, another quantum process algebra was developed: *QPAlg*, due to Lalire and Jorrand[13]. *QPAlg* is very similar to CQP, and it is inspired by the classical process algebras CCS and LOTOS. It does not possess a type system, since it was only conceived to determine how a classical, untyped process formalism can be extended to accommodate quantum phenomena.

There are three kinds of action a process may perform in *QPAlg*:

- communication (using the ! and ? operators);
- unitary transformation; and
- measurement.

The unitary transformations available for direct use in *QPAlg* are the Hadamard gate (H), the controlled–not gate (CNot), the identity (I) and the Pauli transformations $\sigma_x, \sigma_y, \sigma_z$ (written as X, Y, Z respectively). *QPAlg* does not have, in its present form, an operator for explicit probabilistic choice. The *QPAlg* paper contains, as examples of applications, the construction of an EPR pair, the teleportation protocol, and BB84. The BB84 example is reproduced partly in Figure 5.2.

Figure 5.2 A simplified model of BB84 using the QPAlg process algebra. The definitions of the agents **A, B, E** are omitted here for simplicity.

$$
\begin{aligned}
\textbf{Alice} &= [a : \mathsf{Qubit} \,.\, \mathbf{A}[a];\ \mathit{fill}!a \,.\, \mathsf{end}];\ \textbf{Alice} \\
\textbf{Bob} &= [b : \mathsf{Qubit} \,.\, \mathit{empty}?b \,.\, \mathbf{B}[b]];\ \textbf{Bob} \\
\textbf{Eve} &= [e : \mathsf{Qubit},\ f : \mathsf{Qubit} \,.\, \mathit{emptyFlaw}?e \,.\, \mathbf{E}[e, f];\ \mathit{fillFlaw}!f \,.\, \mathsf{end}];\ \textbf{Eve} \\
\textbf{Flaw} &= [u : \mathsf{Qubit},\ v : \mathsf{Qubit} \,.\, \mathit{emptyFlaw}!u \,.\, \mathit{fillFlaw}?v \,.\, \mathsf{end}] \\
\textbf{Channel} &= [x : \mathsf{Qubit},\ y : \mathsf{Qubit} \,.\, \mathit{fill}?x \,.\, \mathbf{Flaw}[x, y];\ \mathit{empty}!y \,.\, \mathsf{end}] \\
\textbf{Protocol} &= (\textbf{Alice} \parallel \textbf{Bob} \parallel \textbf{Eve} \parallel \textbf{Channel}) \setminus \{\mathit{fill}, \mathit{empty}, \mathit{fillFlaw}, \mathit{emptyFlaw}\}
\end{aligned}
$$

5.3 Quantum Logic and its Application to Protocols

Ron van der Meyden and Manas Patra have proposed a modal logic for knowledge and time in quantum protocols[14]. They recognise the fact that, in the literature on quantum computation and information, epistemic locutions of the form

" Alice knows x. "

are frequently encountered; the logical framework which they propose is essentially an attempt to make such informal language precise. Their ultimate objective is to lay the foundations for "epistemic analysis" of quantum cryptographic protocols, and related schemes, using logical methods.

Here, we will state the syntax of the quantum logic and show how it has been used to specify certain properties of the B92 protocol for quantum key distribution.

The quantum logic involves formulas over a set of uninterpreted propositions, `Prop`. A formula in the logic may be a proposition, a conjunction or negation of formulae, or one of the following:

- the form $\Box\phi_1$, which retains the usual temporal meaning ("always, formula ϕ_1 holds");
- the form $init(\phi_1)$, which is true if ϕ_1 holds in the initial state of a protocol;
- the form $K_i^c(\phi_1)$, which means "agent i knows, given her classical bits and observations, that ϕ_1 holds in the current state;
- the form $K_i^q(\phi_1)$, which means "agent i knows, given a set of qubits in her possession, that ϕ_1 holds in the current state.

So, the syntax of formulae, Φ, in the quantum logic is given by the following grammar:

$$\Phi ::= p \mid \phi_1 \mid \phi_2 \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid \Box\phi_1 \mid init(\phi_1) \mid K_i^c(\phi_1) \mid K_i^q(\phi_1)$$

where $p \in$`Prop`. The concept of "knowledge" has two variations in the logic, since it depends on what information is used by a particular agent to decide her actions.

In order to define a property using this logic, a model of the protocol under consideration must be built. The logic assumes that protocols are described as *qubit message passing environments*, which are defined as follows (we have modified the original definition slightly):

Definition 5.1 *A **qubit message passing environment** is an abstract model of the computational setting in a quantum protocol, involving agents and channels for synchronous communication. It is defined as a tuple*

$$\langle n, S, I, Act \rangle$$

where n is the number of agents involved in the system, $S = S^q \times S^c$ is the set of all states that occur in the system, I is the initial state and Act is the set of actions performed by the various agents.

The global state S is partitioned into a set of classical states, S^c, and a set of quantum states, S^q. Clearly S^q is a subset of the Hilbert space $\mathscr{H}$ of dimension 2^N, the vector space inhabited by N qubits. The set of classical states consists of elements of the form $s^q = \langle var, loc, chan, res \rangle$, which include

- classical bit assignments, $var(i) : Var_i \longmapsto \{0,1\}$ (here, Var_i is the set of variable names belonging to agent i).
- qubit location assignments, $loc : [0,N] \mapsto [0,n]$. The value of $loc(x)$ is the name of the agent to which x is attached.
- channel value assignments, $chan : [1..n]^2 \mapsto$`Msg`, where `Msg` is a set of classical messages. If $chan(i,j) = m$ in a particular state, this means that message $m \in$`Msg` has just been transmitted from agent i to agent j.

- measurement result assignments. If $res(i) = (M^i, m_i)$ in a particular state, it means that the measurement operator M^i has been applied to the quantum states in S^q, producing as a classical outcome, the value m_i.

Of interest to us is the fact that, if a quantum protocol is expressed in terms of qubit message passing environments, then the quantum logic can be used to define its properties. We will not delve further in the details of such environments, but it is significant to note that they do ultimately provide a sufficient operational model for quantum protocols. It suffices to say that, *if we were to define a programming language whose operational semantics were based on transitions between qubit message passing environments, a protocol specification in this language could be conveniently matched with a set of properties expressed in the quantum logic.*

It is instructive to show how the logic can be used to describe certain properties of B92 formally. Van der Meyden and Patra treat protocols as functions **P** pertaining to a particular environment.

Definition 5.2 *A **run** $r : \mathbb{N} \mapsto S$ describes a potential evolution of the system, with $r(m)$ representing the global state of the system at time m.*

Definition 5.3 *A **protocol** is a system comprised of specific sets of runs, which are generated by various agents engaging in a particular pattern of behaviour. For agent i, a protocol is defined as a function $\mathbf{P} : \mathscr{O}_i^+ \mapsto Act_i$, where $\mathscr{O}_i^+$ is the set of all observations the agent has made, and Act_i is the set of actions performed by the agent.*

For the purpose of providing an informal example of the logic's use, its creators describe B92 as shown in Figure 5.3. They claim that **P**, the protocol representing the eavesdropping version of B92, satisfies the following properties:

$\Box(b = 1 \Rightarrow k_A^c(a) \wedge k_B^c(a))$

("in successful runs, Alice and Bob come to classically know bit a")

$\Box(b = 1 \Rightarrow \neg k_E^c(a))$

("Eve never comes to know bit a based on classical observations alone")

$\Box(b = 1 \Rightarrow k_E^q(a))$

("If Eve could perform repeatable measurements on the qubit intercepted,
she could come to learn the value of a")

5.4 The qSpec Language

The need to demonstrate correctness formally arises as much with communication protocols as it does in the study of algorithms. After all, good software relies partly on correct implementation, but mostly on the algorithms it uses. With the advent of quantum algorithms, whose implementation relies on the development of a substantial quantum computer, formal analysis

Figure 5.3 A simplified model of the B92 protocol, as used by Van der Meyden and Patra to define the protocol's properties in the quantum logic.

1. **Initial State:** Alice has a single qubit, and a classical bit, a. Bob has two classical bits, a' and b. The bases for the set of quantum states S^q in the system are $\boxplus$ and $\boxtimes$.
2. **Alice flips her bit,** a**.**
   - If $a = 0$, she prepares her qubit in state $|0\rangle$.
   - If $a = 1$, she prepares her qubit in state $|+\rangle$.
3. **Alice transmits her qubit to Bob.**
4. **Bob flips his bit** a'**.**
   - If $a' = 0$, he measures the qubit with basis $\boxplus$.
   - If $a' = 1$, he measures the qubit with basis $\boxtimes$.
5. **If the result of the measurement is either** $|0\rangle$ **or** $|+\rangle$**, Bob sets** $b = 0$**.** Otherwise, he sets $b = 1$.
6. **Bob sends a classical message to Alice stating the value of** b**.**
7. **The run is deemed successful only if** $b = 1$**.**

We write **P** for the eavesdropping version of B92, in environment $\mathscr{E}$, if it prescribes the above behaviour for Alice and Bob, and Eve receives the qubit transmitted as well as Bob's classical message. For details, consult Meyden and Patra (2003).
We use the notation

$$k_i^x(a) \equiv K_i^x(a = 0) \vee K_i^x(a = 1)$$

where $x \in \{c, q\}$, to define the properties of **P**.

and simulation are the only practical avenues of investigation. To this end, several quantum programming languages have been developed, including among others QPL[15], QCL[16], and qGCL[17]. The emphasis in these languages is on quantum *computation;* their constructs are mainly intended for manipulating quantum superpositions and expressing quantum parallelism. For it is this parallelism that makes quantum algorithms so efficient and intrinsically different from their classical counterparts.

What the aforementioned languages lack is the ability to express interaction, communication, or, put otherwise, co–operating computations. A program in one of these languages charts the progress of a single computational process; this is suitable for quantum algorithms, which are usually expressed as isolated computations. However, protocols in quantum communication systems explicitly involve the exchange of information between processes, both classical and quantum–mechanical.

The absence of communication features in traditional programming languages is very common. This is quite justified, since programming languages are designed to express computations. Communication between computational processes was given special attention in the 1980s and resulted in the development of CSP and CCS. Indeed, CSP inspired a parallelised programming language with explicit communication constructs, named OCCAM. Furthermore, PROMELA, conceived originally as a dedicated specification language for communication protocols, was designed by Gerard Holzmann for use with SPIN. There are several more languages that are designed to deal with classical systems involving communication between processes.

The development of specification formalisms for quantum protocols has focused, to date, on process algebras, such as CQP and *QPAlg*, which tackle communication explicitly. While process algebras are valuable formal tools with solid mathematical foundations, they are sometimes regarded as too abstract and less intuitive than concrete programming languages. Nevertheless, process algebras place the emphasis on communication and are therefore well–suited to protocol specification. Process equivalences allow for the development of formal proofs, and there exist software tools for analysing specifications in process algebra. Programming languages, on the other hand, are easier to implement in software and can facilitate system simulation.

In what follows, we describe a programming language, named QSPEC[18], that is designed to serve as a middle ground between the quantum process calculi and the quantum programming languages described previously. QSPEC has a syntax that is inspired by PROMELA and PROBMELA. The language will be discussed informally and examples of its use are given. Then we will describe the underlying operational model, which will be used in future work to supply a formal semantics for the language.

5.4.1 QSPEC BY EXAMPLE

A QSPEC program consists of definitions of processes and channels. A process represents an agent taking part in a quantum protocol, capable of performing local computations and exchanging data with other processes. Data exchange occurs over channels, which are abstractions of communication media. Each classical datum in a QSPEC program belongs to one of the following data types:

- `bit`

- `char`
- `string`
- `int`
- `prob`
- `complex`

These are traditional data types commonly found in programming languages: boolean values, characters, strings of characters, integers, probabilities (real numbers between 0 and 1), complex numbers defined as pairs of real numbers. "Quantum variables" can have one of the following types:

- `space[M]`
- `ket`
- `quantumstate`

Any quantum state belongs to a Hilbert space of dimension M; for qubits, $M = 2$. The datatype `space` is for declaring such spaces. Each Hilbert space is spanned by M basis vectors (`kets`); quantum states can be defined in qSpec either as sums of kets, or via a density matrix. The `quantumstate` type is associated with quantum states, no matter which way they are defined. Arrays of the above types are allowed.

Processes taking part in a particular protocol manipulate local data, and can use the `!` operator for transmitting data to channels, as well as the `?` operator for receiving data. The manipulations possible on qSpec data values are:

- boolean operations: conjunction (`and{b,c}`), disjunction (`or{b,c}`) and inversion (`not{b}`);
- arithmetic for integers, probabilities and complex numbers, including modulo n calculations and absolute value;
- concatenation of strings and characters (e.g. `"ele"^"ment"`);
- variable assignment (e.g. `a:=2;`);
- preparation of a quantum state, using a linear combination or tensor product of kets, or through a density matrix;
- unitary transformations on quantum states;
- projective measurement of a single quantum state;

Below is a simple example of qSpec code that illustrates the syntax.

```
process Example {
complex c := (1.0,2.5);
space[2] qubits;
ket |0>, |1>;
basis b of qubits := {{ |0>, |1> }};
quantumstate[b] qu1 ::== sqrt(0.5)*|0> + sqrt(0.5)*|1>;
quantumstate[b] qu2 ::== sqrt(0.3)*|0> + sqrt(0.7)*|1>;
quantumstate[b] qu3 ::== tprod(qu1,qu2);
space[4] hilbert4;
ket |0>,|1>,|2>,|3>;
basis b4 of hilbert4 := {{ |0>,|1>,|2>,|3> }};
quantumstate[b4] onlythree ::== 0*|0> + sqrt(1/3)*|1> + sqrt(1/3)*|2> + sqrt(1/3)*|3>;
quantumstate[b4] result;
result ::== measure(onlythree);
}
```

Next is an extended example of qSpec, that models the quantum transmission phase of BB84.

```
global const N=10;
space[2] qubits;
basis bR of qubits := {{ |0>, |1> }};
basis bD of qubits := {{ |2>, |3> }};
chan(Alice,Bob) of quantumstate[qubits] opfibre;
chan(Alice,Bob) of string[1] a2b;
chan(Alice,Bob) of string[1] b2a;
process Alice {
array[N] of bit randombits;
array[N] of bit prepbases; // 0 for rectilinear basis, 1 for diagonal
int i=0;
while (i<N) {
   probabilistic {
     0.5 -> randombits[i]:=0;
     0.5 -> randombits[i]:=1;
   }
   bit val := randombits[i];
   probabilistic {
     0.5 -> quantumstate[bR] photon ::== abs(val-1)*|0> + abs(val)*|1>; prepbases[i]:=0;
     0.5 -> quantumstate[bD] photon ::== abs(val-1)*|2> + abs(val)*|3>; prepbases[i]:=1;
   }
   opfibre!photon;
   string ack;
   b2a?ack;
   if
   :: (ack=="ready") -> i:=i+1;
   :: (ack!="ready") -> skip;
```

```
   fi;
}}
process Bob {
array[N] of bit basischoices;
array[N] of bit measoutcomes;
int i=0;
while (i<N) {
   probabilistic {
      0.5 -> basischoices[i]:=0; quantumstate[bR] measurementresult;
      0.5 -> basischoices[i]:=1; quantumstate[bD] measurementresult;
   }
   opfibre?measure(measurementresult);
   b2a!ack;
}}
```

5.4.2 Understanding qSpec and its Operational Model

The objective of a qSpec program is to specify the behaviour of a set of processes capable of manipulating both classical and quantum data. The most elementary notions in qSpec are that of a *process* and a *channel*. A *process* is understood as an agent that performs a sequential computation on classical and quantum data, which is capable of transmitting and receiving data from other processes. A *channel* in qSpec is a queue of data available for asynchronous communication between processes; classical channels store ordinary data values (such as integers and bits), while quantum channels store quantum states.

A protocol specification in qSpec consists, therefore, of declarations of processes and declarations of channels. Every process has its own local variables, and channels can only be used by selected processes. In particular, the user must specify to which processes a given channel is available. Thus, there are no global variables in qSpec.

The operational model of any qSpec program consists of three elements: a set of processes, P, a set of classical channels, C, and a set of quantum channels, Q. Hence, $Systemstate = \langle P, C, Q \rangle$ denotes the overall system state at any given moment.

A process has a state at any given time which depends on the contents of a classical store of variables, a quantum store, and a set of channel names. In particular, the classical store contains the names of classical variables used in the process, their types and their values. The quantum store, similarly, contains the names of quantum variables used in the process and their values, while the names of channels available for sending and receiving to the process are contained in a set. Formally:

Definition 5.4 *A process in* qSpec *is modelled by a tuple*

$$(\rho, \sigma, \text{CN})$$

where ρ is a classical store, σ is a quantum store, and CN *is a set of channel names available to the process for communication.*

Definition 5.5 *A classical store is modelled by a function*

$$\rho : CVar \mapsto Type \times CValue$$

which associates with each classical variable a type and a value.

Definition 5.6 *A quantum store is modelled by a function*

$$\sigma : QVar \mapsto QValue$$

which associates with each quantum variable a quantum state, belonging to a Hilbert space of suitable dimensions.

A classical channel is capable of storing classical variables and their values. In qSpec, when a process sends a variable to a classical channel, a complete copy of the variable, its type and value is introduced to the channel store, assuming that the process has access to that channel. Each channel has a label l. Any process that requires access to a channel with label l must have l in its set of channel names cn. Formally a classical channel is defined as follows:

Definition 5.7 *A classical channel is a tuple*

$$(l, \rho^{CH})$$

where l is a unique label or channel name, and ρ^{CH} is a classical store.

Condition 5.1 *We stipulate that a process $(\rho, \sigma, \text{cn})$ can only use classical channel (l, ρ^{CH}) if and only if $l \in$ cn.*

Quantum channels are similarly modelled by tuples of the form (λ, σ^{CH}) with λ being a unique channel label and σ^{CH} being a quantum store. The operational semantics of a qSpec program is given by rules stating transitions between system states. The full definition of these and typing rules is an area for future work.

5.5 Quantum System Simulation

In this section we will review the work of Black and Lane (2004), who have modelled BB84 with the *QCSim* simulator and computed the probability, for this protocol, that information is received faithfully.

Black and Lane take an intermediate approach to the simulation and verification techniques we have presented. They propose a "simulator for quantum information systems which, like for the simulation mode of spin, executes a trial of a particular model by resolving all non–deterministic choices, while at the same time providing feedback to the user. Unlike spin simulations, the *QCSim* tool considers all possible executions of a model, as would spin in verification mode (or indeed any model checker). Thus, *QCSim* is a simulator which includes features typically found only in verification tools. Before discussing the capabilities of *QCSim*, it is worthwhile to consider some of the issues surrounding the simulation of quantum communication schemes.

According to the authors just cited, when simulating quantum information systems, it suffices to assume discrete time steps and Hilbert spaces of finite dimensions. Also, it is useful to make provisions for "extra–systemic processing", including extra qubits, gates and operations —beyond those present in the system being modelled. Since a simulator is "a largely fixed piece of software," it cannot handle with equal efficiency all the problems one would want to consider. Thus, although it seems convenient to analyse BB84 with *QCSim*, this may not be so for all quantum protocols.

A *QCSim* model starts with declarations of all qubits used in a particular quantum simulation. Then the initial state of the whole system is defined, and the remainder of the model consists of quantum gate operations and commands. Note that *QCSim* is capable of handling *mixed quantum states* (these are not expressible in terms of Dirac ket vectors). Using mixed states, noise processes and quantum decoherence[19] can be modelled directly.

Black and Lane have reduced BB84 to a quantum circuit, using only quantum gates and measurements. The limitation of their approach is that it fails to model communication explicitly; all aspects of a quantum protocol are reduced to computational stages in a circuit. Therefore, we take the view here that their approach is overly restrictive for a systematic analysis of other quantum protocols.

5.6 More on Probabilistic Model Checking

As we demonstrated in Chapter 4, probabilistic model checking is definitely a viable and relevant technique for modelling quantum protocols. While the PRISM tool has been under development for several years and has been applied effectively to numerous problems, its state–based input language is counterintuitive for someone accustomed to programming in the style of C, Pascal and other imperative languages. The recent proposal by Baier et al. (2004), of an imperative modelling language for "communicating probabilistic processes" seems like an excellent remedy; the language itself is named PROBMELA, in recognition of its similarities with PROMELA and the fact that it explicitly models probability. The core of PROBMELA features deterministic and randomised assignments, communication, conditional and repetitive statements as well as guarded commands. The core has been extended by the creators of the language with channel–based message passing and atomic regions.

The complete syntax of PROBMELA commands is given by the following grammar:

$$
\begin{aligned}
P ::= {} & \mathsf{skip} \mid x := expr \mid x := \mathsf{random}(V) \mid d?x \mid d!expr \mid P_1; P_2 \\
& \mid \mathsf{IF} \ :: g_1 \Rightarrow P_1 \ \cdots \ :: g_n \Rightarrow P_n \ \ \mathsf{FI} \\
& \mid \mathsf{DO} \ :: g_1 \Rightarrow P_1 \ \cdots \ :: g_n \Rightarrow P_n \ \ \mathsf{OD} \\
& \mid \mathsf{PIF} \ \ [p_1] \Rightarrow P_1 \ \cdots \ \ [p_n] \Rightarrow P_n \ \ \mathsf{FIP} \\
& \mid \mathsf{atomic}\{P\}
\end{aligned}
$$

The semantics of PROBMELA is expressed in terms of Markov decision processes, which were discussed in section 3.5.3. This language has been used to describe the `IPv4 zeroconf` communication protocol[20], and we feel that it is highly suitable for describing the random behaviour exhibited within quantum protocols.

To conclude this section, let it be noted that Frank Ciesinski and others have developed a probabilistic model checker, *probUSM*[21], which uses PROBMELA as its description language. We express the hope that continued development of *probUSM* may provide an alternative platform to PRISM for studying quantum protocols.

5.7 Summary

We have presented, in this chapter, several alternative approaches to the task of modelling and analysing quantum protocols. We discussed the two quantum process algebras, CQP and *QPAlg*, and saw two examples of their use. We studied the logic of knowledge and time for quantum systems from Meyden and Patra (2003) in detail, and criticised the *QCSim* simulator of Black and Lane (2004). Finally, we directed our attention to alternative means of performing probabilistic model checking.

6

Recapitulation and Directions for Future Work

This thesis is the culmination of a single year's research on the application of formal methods to the design and validation of protocols for quantum cryptography. In order to conclude the presentation of this work, we will now review the salient points of the thesis. The final section of this chapter will focus on current trends in computer science research with regard to quantum protocols.

6.1 Review and Valuation of this Work

We can summarise the main achievements of our work as follows:

- We have considered "quantum protocols," including quantum key distribution, quantum coin flipping, quantum bit commitment, quantum teleportation and dense coding *as a distinct, novel class of communication schemes.* Of these, we have discussed in detail only the BB84, B92 and E91 variants of quantum key distribution.
- We have detailed the technique of model checking, and demonstrated its use as an alternative means of investigating the properties of such protocols. In particular, *we have shown that probabilistic model checking can provide a practical alternative to information–theoretic proofs of security.*
- We have discussed the use of *quantum process algebras* and their applicability in the context of quantum protocols.
- *Logical methods for defining the properties of quantum protocols,* including a dedicated "quantum logic," have been presented.

On a per chapter basis, the topics described in this thesis have been:

- **Chapter 2**
  - the fundamental aspects of quantum theory relevant to quantum protocols, including orthogonality of quantum states and entanglement;
  - Shannon's formalisation of the concept of perfect security;
  - the importance of secret–key reconciliation and privacy amplification; and
  - Dominic Mayers' security criteria for quantum key distribution.
- **Chapter 3**
  - the principle of model checking;
  - several specification languages for system and property descriptions;
  - the distinction between linear temporal logic and branching temporal logic; and
  - how the measurement of an EPR pair can be formalised using random events and logical formulae.
- **Chapter 4**
  - what types of attack an eavesdropper may attempt in an attempt to subvert a quantum cryptographic protocol;
  - how PRISM has been used to compute the probability of detecting an eavesdropper in BB84, and the probability of an eavesdropper obtaining a significant number of accurate measurements in the same protocol; and
  - how SPIN might be used to show that BB84 satisfies the criterion for security against tampering.
- **Chapter 5**
  - the characteristics of the quantum process algebras CQP and *QPAlg;*
  - the logic, due to Ron van der Meyden and Manas Patra, for knowledge and time in quantum systems; and
  - the probabilistic specification language PROBMELA, and the model checker *probUSM*.

We have reason to believe that the various techniques presented herein are likely to become increasingly relevant to physicists as well as other potential designers and implementors of quantum protocols. The ability to develop and test parameterised models of such protocols, as we have done with PRISM for BB84, provides a useful way of describing implementation–specific features. For instance, in a particular implementation of BB84, the probability P_L of an eavesdropper making a correct measurement, despite an incorrect choice of basis, may depend on the physical limitations of some measurement device. This can be incorporated easily in a system description suitable for model checking.

While practical developments in the area of quantum information are necessarily driven by what is possible experimentally, computer scientists are able to contribute significantly to the theoretical foundations and design of future systems. Well–established techniques in computing, such as those discussed here, have more to provide to the field than meets the eye. As we mentioned in the first chapter, this has been the case for classical communication protocols, and especially so in the cases where security is a foremost concern.

6.2 Open Issues and Trends

One of the questions that need to be addressed, before any study of quantum protocols can be said complete, is: "What is the smallest set of primitive operations with which all quantum protocols can be described?" Clearly, this cannot be answered without recourse to physics; experimental physicists are likely to discover phenomena we are currently unaware of, and these might be put to use in new protocols. Thus, we believe that current research in formal models and semantics for quantum protocols is still at an early stage; it will necessarily be driven by developments in physics.

From our investigations, we have concluded that all quantum protocols proposed to date are generally interleavings of quantum computations, classical computations, and communication over (mainly) quantum channels. The most promising feature of quantum theory with regard to quantum protocols is entanglement, and there is a growing interest in the potential of this phenomenon. We have not discussed entanglement–based protocols in detail in this thesis; clearly, this is a specific area for future investigations.

An important point is the fact that, while the qubit has been regarded as a basic "unit" in most of the quantum computing and information research, there exist quantum systems of higher dimensions. Fitzgerald's recent dissertation[1] highlights the importance of this by dealing specifically with *qudits*, i.e. quantum states with d basis vectors. Qudits are likely to be useful for developing new quantum protocols: they could be used, for instance, to extend the dense coding protocol so that more than two classical bits may be encoded onto a single quantum system. One of the distinctive features of qSpec is that it allows for the manipulation of qudits, and we believe that other quantum programming languages will soon incorporate this facility.

6.3 Conclusion

We feel that standard techniques in computer science for formal specification and verification are valuable means for understanding and analysing quantum protocols; it is our hope that we have demonstrated this successfully in this thesis. If the methods we have described do indeed prove useful to the quantum information community, our research will have achieved its primary goal.

7

Appendix: SPIN Simulation Model of BB84

```
#define INITIALKEYLENGTH 10

/*** Declarations of data types ***/
mtype = {ack, ok, nok};
typedef particle {
bit basis;
bit value;
}
typedef tuple {
int index;
bit chosenbasis;
}

/*** Communication channels ***/
chan opfibre = [1] of {particle};   /* quantum channel */
chan a2b = [1] of {mtype};   /* classical channel from Alice to Bob */
chan b2a = [1] of {tuple};   /* classical channel from Bob to Alice */

/*** Inline function definition for measuring photon p with basis b ***/
inline decode(p,b) {
   bit measureresult=0;
   /* need to compare p.basis with b */
   if
   :: (p.basis==b) -> printf("Correct basis."); measureresult = p.value;
   :: (p.basis!=b) -> measureresult = 0;
   :: (p.basis!=b) -> measureresult = 1;
   fi;
   printf("Measurement result = %d\n", measureresult);
};
```

```
/*** Sender (Alice) ***/
active proctype Alice()
{
   bit initialkey[INITIALKEYLENGTH];
   bit encodingbases[INITIALKEYLENGTH];
   int a_counter=0;

   /* Preparation phase: */
   atomic {
   do
   :: (a_counter<INITIALKEYLENGTH) -> initialkey[a_counter] = 0; a_counter++;
   :: (a_counter<INITIALKEYLENGTH) -> initialkey[a_counter] = 1; a_counter++;
   :: else -> break;
   od;
   a_counter=0;
   do
   :: (a_counter<INITIALKEYLENGTH) -> encodingbases[a_counter] = 0; a_counter++;
   :: (a_counter<INITIALKEYLENGTH) -> encodingbases[a_counter] = 1; a_counter++;
   :: else -> break;
   od;
   };  /* end atomic construct */

   /* Quantum Transmission phase: */
   a_counter = 0;
   do
   :: (a_counter<INITIALKEYLENGTH) ->
      particle photon;
      photon.basis = encodingbases[a_counter];
      photon.value = initialkey[a_counter];
      printf("Alice sent photon representing the bit value %d\n", photon.value);
      opfibre!photon.basis;
      opfibre!photon.value;
      b2a?ack;
      a_counter++;
   :: else -> break;
   od;

   /* Public Discussion phase */
   tuple decoding;
   do
   :: b2a?decoding.index -> b2a?decoding.chosenbasis;
      if
```

```
      :: (encodingbases[decoding.index] == decoding.chosenbasis) -> a2b!ok;
      :: (encodingbases[decoding.index] != decoding.chosenbasis) -> a2b!nok;
      fi;
   od;
}

/*** Receiver (Bob) ***/
active proctype Bob()
{
   bit decodingbases[INITIALKEYLENGTH];
   bit measurements[INITIALKEYLENGTH];
   int b_counter=0;
   particle recvd;

   /* Choosing bases for measurements */
   atomic {
   do
   :: (b_counter<INITIALKEYLENGTH) -> decodingbases[b_counter] = 0; b_counter++;
   :: (b_counter<INITIALKEYLENGTH) -> decodingbases[b_counter] = 1; b_counter++;
   :: else -> break;
   od;
   };  /* end atomic construct */

   /* Receiving Photons and Making Measurements */
   b_counter = 0;
   do
   :: opfibre?recvd.basis -> opfibre?recvd.value;
      atomic {
      decode(recvd, decodingbases[b_counter]);
      measurements[b_counter] = measureresult;
      };  /* end atomic construct */
      b_counter++;
      b2a!ack;
   od;

   /* Public Discussion phase */
   /* Bob sends tuples of the form (index,basis) to Alice, to tell her what bases
   he chose for measuring each photon: */
   b_counter = 0;
   do
   :: (b_counter<INITIALKEYLENGTH) ->
      tuple tup;
      tup.index = b_counter;
```

```
      tup.chosenbasis = decodingbases[b_counter];
      b2a!tup.index;
      b2a!tup.chosenbasis;
      /* waiting for Alice's reply and react: */
      do
      :: a2b?ok -> printf("Bob used the CORRECT basis for measuring this photon.");
      :: a2b?nok -> printf("Bob used the WRONG basis for measuring this photon.");
      od;
      b_counter++;
   :: else -> break;
   od;
};

/*** Eavesdropper (Eve) ***/
active proctype Eve()
{
   bit evedecodingbases[INITIALKEYLENGTH];
   bit evemeasurements[INITIALKEYLENGTH];
   int e_counter=0;
   particle everecvd;

   /* Choosing bases for measurements */
   atomic {
   do
   :: (e_counter<INITIALKEYLENGTH) -> evedecodingbases[e_counter] = 0; e_counter++;
   :: (e_counter<INITIALKEYLENGTH) -> evedecodingbases[b_counter] = 1; e_counter++;
   :: else -> break;
   od;
   };  /* end atomic construct */

   /* Receiving Photons and Making Measurements */
   e_counter = 0;
   do
   :: opfibre?everecvd.basis -> opfibre?everecvd.value;
      opfibre!everecvd.basis; opfibre!everecvd.value; /* intercept-resend attack */
      atomic {
      decode(everecvd, decodingbases[e_counter]);
      measurements[e_counter] = measureresult;
      };  /* end atomic construct */
      e_counter++;
   od;
}
```

Notes

Preface:

[1]See Bill Gates, *Microsoft Progress Report: Security,* March 31, 2004; available from `http://www.microsoft.com/mscorp/execmail/2004/03-31security.asp` [visited August 6, 2004].

[2]See Holzmann (1991).

[3]See Ryan et al. (2001).

[4]See Holzmann (1991) *loc. cit*.

[5]See `http://www.ftponline.com/special/testing/holzmann` for a transcript of this interview [visited August 6, 2004].

[6]See Feynman (1999).

[7]See Williams and Clearwater (2000).

[8]See Gruska (1999); Nielsen and Chuang (2000); Williams and Clearwater (2000).

[9]See Stewart (1997).

[10]See Landauer (1999).

[11]See Holzmann (2003).

[12]See Kwiatkowska et al. (2004); Parker et al. (2004).

[13]See Bennett and Brassard (1984).

[14]See Mayers (2001).

[15]*Ibid.*

[16]See Papanikolaou (2004).

[17]See Nagarajan and Gay (2004).

[18]See Baier et al. (2004).

[19]See Meyden and Patra (2003).

[20]See Black and Lane (2004).

Chapter 1:

[1]See Gruska (1999); Nielsen and Chuang (2000).

[2]See Landauer (1999).

[3]See Gruska (1999); Nielsen and Chuang (2000).

[4]See Wiesner (1969).

[5]See Mayers (2001).

[6]Note that implementations which include imperfect apparatus necessarily have limited security; for instance, some of the original prototypes of quantum cryptography were known to produce noises sufficiently discernible to facilitate the enemy's task greatly.

[7]Practical systems for quantum cryptography are already commercially available at the time of writing from the companies NEC, MAGIQ, and ID QUANTIQUE.

[8]Note that δ_{xy} is the Dirac delta function, which takes the value 1 if $x = y$, or 0 otherwise.

[9]See Wiesner (1969).

[10]See Wootters and Zurek (1982).

[11]A matrix $\mathbf{U}$ is *unitary* if it is equal to its conjugate transpose, or *adjoint*: $\mathbf{U} = \mathbf{U}^\dagger = (\mathbf{U}^*)^T$.

[12]See Brassard (1988).

[13]This is indeed what quantum theory predicts: measurement is destructive in the sense that a quantum state is changed, and changed *permanently.*

[14]See Diffie and Hellman (1976).

[15]See Welsh (1998).

[16]See Brassard and Crépeau (1991); Bennett et al. (1992); Brassard et al. (1993).

[17]See Schneier (1996).

[18]*Ibid.*

[19]See Brassard et al. (1993); Brassard and Crépeau (1991).

[20]See Mayers (1996).

[21]See Milner (1989).

[22]See Hoare (1985).

[23]See Mauw and Veltink (1993).

[24]See Holzmann (1991, 2003).

[25]See Baier et al. (2004).

CHAPTER 2:

[1]See Bennett and Brassard (1984).

[2]See Bennett (1992).

[3]See Ekert (1991).

[4]See Brassard and Salvail (1994).

[5]See Bennett et al. (1988).

[6]See Brassard and Crépeau (1991); Brassard et al. (1993).

[7]See Mayers (1996).

[8]See Wiesner (1969); Bennett and Wiesner (1992).

[9]See Bennett and Wiesner (1992).

[10]See Bennett et al. (1993).

[11]See Cohen-Tannoudji et al. (1977).

[12]See Shannon (1949).

[13]*Ibid.*

[14]See Wolf (1999); Smart (2003).

[15]See Gruska (1999).

[16]*Ibid.*

[17]See Bennett and Brassard (1984).

[18]See **Bennett (1992)**.

[19]See **Ekert (1991)**.

[20]See Bouwmeester et al. (2000); Gruska (1999).

[21]See Gruska (1999); Nielsen and Chuang (2000).

[22]See Ekert (1991).

[23]See Bouwmeester et al. (2000).

[24]See Brassard and Salvail (1994).

[25]*Ibid.*

[26]See Bennett et al. (1988).

[27]See Wegman and Carter (1981).

[28]See Elliot (2004).

[29]See Bennett et al. (1988).
[30]See Yamamoto (2004).
[31]See Wegman and Carter (1981).
[32]See Mayers (2001).
[33]See Bennett and Wiesner (1992).

CHAPTER 3:

[1]See Huth and Ryan (2000).
[2]See Holzmann (2003).
[3]See Kwiatkowska et al. (2004).
[4]See Hoare (1985).
[5]See Ryan et al. (2001).
[6]*Ibid.*
[7]See Milner (1999).
[8]See Gay and Nagarajan (2002).
[9]See Nagarajan and Gay (2004).
[10]See Holzmann (1991).
[11]See Holzmann (2003) *loc. cit*.
[12]See Holzmann (1991) *loc. cit*.
[13]See Baier et al. (2004).
[14]See Papanikolaou (2004).
[15]See Holzmann (2003) *loc. cit*.
[16]The reason for delving into this issue is that the **SPIN** and **PRISM** tools, used for analysis of quantum protocols in this work, use entirely different logics for property specification.
[17]See Vardi (2001).
[18]*Ibid.*
[19]See Holzmann (2003) *loc. cit*.
[20]*Ibid.*
[21]See Kwiatkowska et al. *loc. cit*.
[22]See Holzmann (2003, pp. 137—138) *loc. cit*.
[23]See Baier et al. (1997, 2004).
[24]See Jonsson et al. (2001).
[25]See Nelson (1995).
[26]See Monniaux (2004).
[27]See Emerson (1990).
[28]See Ciesinski and Größer (2004).
[29]See Bub (2002).
[30]Note that the symbols $\psi_1, \ldots \psi_4$ are used instead of the full ket vectors they denote to lighten the notation.
[31]See Cohen-Tannoudji et al. (1977).
[32]See Caves et al. (2002).
[33]See Parker et al. (2004).

CHAPTER 4:

[1]See Gay and Nagarajan (2002); Nagarajan and Gay (2004).
[2]See Yamamoto (2004).
[3]This algorithm is available for use directly within the plotting program GnuPlot.
[4]See Holzmann (2003).
[5]See Gruska (1999); Mayers (2001).

CHAPTER 5:

[1]See Bérard et al. (1999); Ryan et al. (2001); Holzmann (1991, 2003); Mauw and Veltink (1993).
[2]The model checker in question is **MCK** ("Model Checking Knowledge").

[3]See Hoare (1985).
[4]See Milner (1989).
[5]See Ponse et al. (2001).
[6]See Black and Lane (2004).
[7]See Papanikolaou (2004).
[8]See Nagarajan and Gay (2004).
[9]See Milner (1999).
[10]See Nagarajan and Gay (2004).
[11]Private communication.
[12]See Nagarajan and Gay (2004).
[13]See Lalire and Jorrand (2004).
[14]See Meyden and Patra (2003).
[15]See Selinger (2004).
[16]See Ömer (1998).
[17]See Sanders and Zuliani (2000).
[18]See Papanikolaou (2004).
[19]See Gruska (1999).
[20]See Baier et al. (2004).
[21]See `http://web.informatik.uni-bonn.de/I/baier/projectpages/PROBUSM/`.

CHAPTER 6:

[1]See Fitzgerald (2004).

Bibliography

Abadi, M. and A. Gordon (1999). A calculus for cryptographic protocols: The spi–calculus. *Information and Computation 148*, 1–70.

Baier, C., F. Ciesinski, and M. Größer (2004). PROBMELA: A modeling language for communicating probabilistic processes. In *Proc.* MEMOCODE *04*. IEEE CS Press.

Baier, C., E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan (1997). Symbolic model checking for probabilistic processes. In *Proc. 24th International Colloquium on Automata, Languages and Programming (ICALP'97)*, Volume 1256 of *Lecture Notes in Computer Science*, pp. 430–440.

Bennett, C. (1992). Quantum cryptography using any two nonorthogonal states. *Physical Review Letters 68*(21), 3121—3124.

Bennett, C., G. Brassard, S. Breidbart, and S. Wiesner (1982). Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology — Proceedings of* CRYPTO *'82*. Plenum Press.

Bennett, C. H. and G. Brassard (1984, December). Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of International Conference on Computers, Systems and Signal Processing*.

Bennett, C. H., G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. K. Wootters (1993). Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channels. *Physical Review Letters 70*, 1895–1899.

Bennett, C. H., G. Brassard, C. Crépeau, and M.-H. Skubiszewska (1992). Practical quantum oblivious transfer. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pp. 351–366. Springer–Verlag.

Bennett, C. H., G. Brassard, and J.-M. Robert (1988). Privacy amplification by public discussion. SIAM *J. Comput. 17*(2), 210–229.

Bennett, C. H. and S. J. Wiesner (1992). Communication via one- and two-particle operators on Einstein–Podolsky–Rosen states. *Physical Review Letters 69*(20), 2881—2884.

Bérard, B., M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, P. Schnoebelen, and P. McKenzie (1999). *Systems and Software Verification: Model–Checking Techniques and Tools*. Springer–Verlag. Updated version of the French language edition: "Verification de logiciels. Techniques et outils du model–checking" coordonné par Philippe Schnoebelen, Copyright Vuibert, Paris, 1999.

Black, P. E. and A. W. Lane (2004). Modeling quantum information systems. Unpublished.

Bouwmeester, D., A. Ekert, and A. Zeilinger (Eds.) (2000). *The Physics of Quantum Information*. Springer–Verlag.

Brassard, G. (1988). *Modern Cryptology: A Tutorial*, Volume 325 of *Lecture Notes in Computer Science*. Springer–Verlag.

Brassard, G. and C. Crépeau (1991). Quantum bit commitment and coin tossing protocols. In A. Menezes and S. Vanstone (Eds.), *Advances in Cryptology — CRYPTO '90*, pp. 49—61. Springer–Verlag. Volume 537 of *Lecture Notes in Computer Science*.

Brassard, G., C. Crépeau, R. Josza, and D. Langlois (1993). A quantum bit commitment scheme provably unbreakable by both parties. In *Proceedings of 34th Annual IEEE Symposium on the Foundations of Computer Science*, pp. 362—371.

Brassard, G. and L. Salvail (1994). Secret-key reconciliation by public discussion. In *Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT '93)*, pp. 410–423. Springer–Verlag.

Bub, J. (2002). Quantum entanglement and information. In The Stanford Encyclopedia of Philosophy, available at `http://plato.stanford.edu/archives/win2002/entries/qt-entangle/`.

Caves, C. M., C. A. Fuchs, and R. Schack (2002). Quantum probabilities as Bayesian probabilities. *Physical Review A 65*, 022305–1 to 022305–6.

Ciesinski, F. and M. Größer (2004). On Probabilistic Computation Tree Logic. To appear.

Cohen-Tannoudji, C., B. Diu, and F. Laloë (1977). *Quantum Mechanics*, Volume I. Wiley–Interscience.

Diffie, W. and M. E. Hellman (1976). New directions in cryptography. IEEE *Transactions on Information Theory IT–22*(6), 644—654.

Ekert, A. (1991). Quantum cryptography based on Bell's theorem. *Physical Review Letters 67*(6), 661—663.

Elliot, C. (2004, July/August). Quantum cryptography. IEEE *Security & Privacy Magazine 2*(4), 57—61.

Emerson, E. A. (1990). Temporal and modal logic. Volume B: Formal Models and Semantics, pp. 995–1072. MIT Press.

Feynman, R. P. (1999). There's plenty of room at the bottom. In A. Hey (Ed.), *Feynman and Computation: Exploring the Limits of Computers*, Chapter 7, pp. 63—76. Perseus Books.

Fitzgerald, D. (2004). Quantum qudit simulation. Master's thesis, National University of Ireland, Galway. M.Sc. in Software Design and Development, Supervisor: Dr. Michael McGettrick.

Gay, S. and R. Nagarajan (2002). Formal verification of quantum protocols. Available from e–print archive arXiv.org (record: quant-ph/0203086).

Gisin, N., G. Ribordy, W. Tittel, and H. Zbinden (2002). Quantum cryptography. *Reviews of Modern Physics 74*(1), 145—195.

Gruska, J. (1999). *Quantum Computing*. McGraw–Hill International.

Hailpern, B. T. (1982). *Verifying Concurrent Processes Using Temporal Logic*, Volume 129 of *Lecture Notes in Computer Science*. Springer–Verlag.

Hamming, R. (1986). *Coding and Information Theory* (2nd ed.). Prentice–Hall.

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice–Hall.

Holzmann, G. (1991). *The Design and Validation of Computer Protocols*. Prentice–Hall.

Holzmann, G. (2003). *The SPIN Model Checker: Primer and Reference Manual*. Pearson Education.

Huth, M. R. and M. D. Ryan (2000). *Logic in Computer Science: Modelling and Reasoning About Systems* (1st ed.). Cambridge University Press.

Jonsson, B., W. Yi, and K. G. Larsen (2001). Probabilistic extensions of process algebras. In J. A. Bergstra, A. Ponse, and S. A. Smolka (Eds.), *Handbook of Process Algebra*. Elsevier Science.

Kwiatkowska, M., G. Norman, and D. Parker (2004). Modelling and verification of probabilistic systems. In P. Panangaden and F. V. Breugel (Eds.), *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*. American Mathematical Society. Volume 23 of CRM Monograph Series.

Lalire, M. and P. Jorrand (2004, June). A process algebraic approach to concurrent and distributed quantum computation: Operational semantics. In *Proceedings of The 2nd International Workshop on Quantum Programming Languages*, pp. 109—126. Turku Centre for Computer Science (TUCS).

Landauer, R. (1999). Information is inevitably physical. In A. Hey (Ed.), *Feynman and Computation: Exploring the Limits of Computers*. Perseus Books.

Lo, H.-K. (2003). Method for decoupling error correction from privacy amplification. *New Journal of Physics 5*, 36.1—36.24.

Lomonaco, Jr., S. J. (1998). A quick glance at quantum cryptography. Available from e–print archive arXiv.org (record: quant-ph/9811056).

Mauw, S. and G. Veltink (1993). *Algebraic Specification of Communication Protocols*, Volume 36 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Mayers, D. (1996). Unconditionally secure quantum bit commitment is impossible. In *Fourth Workshop on Physics and Computation — PhysComp '96*. Springer–Verlag.

Mayers, D. (2001). Unconditional security in quantum cryptography. *Journal of the* ACM *48*(3), 351—406.

Meyden, R. and M. Patra (2003). Knowledge in quantum systems (Extended Abstract). In *Proc. Conf. On Theoretical Aspects of Rationality and Knowledge*, pp. 104–117. ACM Press.

Milner, R. (1989). *Communication and Concurrency*. Prentice–Hall.

Milner, R. (1999). *Communicating and Mobile Systems: The π–Calculus*. Cambridge University Press.

Molotkov, S. N. (2004). Integration of quantum cryptography into fiber-optic telecommunication systems. JETP *Letters 79*(11), 559—570.

Monniaux, D. (2004). Abstract interpretation of programs as Markov decision processes. Available at `http://www.di.ens.fr/~monniaux`.

Nagarajan, R. and S. Gay (2004, June). Communicating quantum processes. In *Proceedings of The 2nd International Workshop on Quantum Programming Languages*. Turku Centre for Computer Science (TUCS).

Nelson, B. L. (1995). *Stochastic Modeling: Analysis and Simulation*. McGraw–Hill.

Nielsen, M. A. and I. L. Chuang (2000). *Quantum Computation and Quantum Information*. Cambridge University Press.

Ömer, B. (1998). A procedural formalism for quantum computing. Master's thesis, Department of Theoretical Physics, University of Vienna.

Papanikolaou, N. (2004). QSPEC: A programming language for quantum communication systems design. In *Proceedings of PREP2004 Postgraduate Research Conference in Electronics, Photonics, Communications & Networks, and Computing Science*. EPSRC.

Parker, D., G. Norman, and M. Kwiatkowska (2004, February). PRISM 2.0 users' guide. Available at `http://www.cs.bham.ac.uk/~dxp/prism`.

Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th* IEEE *Symposium on Foundations of Computer Science*. IEEE Press.

Ponse, A., S. A. Smolka, and J. A. Bergstra (2001). *Handbook of Process Algebra*. Elsevier Science Inc.

Rescher, N. and A. Urquhart (1971). *Temporal Logic*, Volume 3 of *Library of Exact Philosophy*. Springer–Verlag.

Rieffel, E. and W. Polak (2000). An introduction to quantum computing for non–physicists. ACM *Computing Surveys 32*(3), 300—335.

Ryan, P., S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe (2001). *Modelling and Analysis of Security Protocols*. Pearson Education.

Sanders, J. and P. Zuliani (2000). Quantum programming. In *Mathematics of Program Construction*, pp. 80—99. Springer. Volume 1837 of *Lecture Notes in Computer Science*.

Schneier, B. (1996). *Applied Cryptography* (2nd ed.). Wiley.

Selinger, P. (2004). Towards a quantum programming language. *Mathematical Structures in Computer Science 14*(4), 527–586.

Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell Systems Technical Journal 28*(4), 656—715.

Smart, N. (2003). *Cryptography: An Introduction*. McGraw–Hill Education (UK).

Stewart, I. (1997). *Does God Play Dice?* Penguin Books.

Stirling, C. (2001). *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer–Verlag.

Van der Meyden, R. and M. Patra (2003). A logic for probability in quantum systems. In M. Baaz and J. A. Makowsky (Eds.), *Computer Science Logic, 17th International Workshop,* CSL *2003, 12th Annual Conference of the* EACSL*, and 8th Kurt Gödel Colloquium,* KGC *2003, Vienna, Austria, August 25-30, 2003, Proceedings*, Volume 2803 of *Lecture Notes in Computer Science*, pp. 427—440. Springer.

Vardi, M. Y. (2001). Branching vs. linear time: Final showdown. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 1–22. Springer–Verlag.

Wegman, M. and J. Carter (1981). New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci. 22*, 265—279.

Welsh, D. (1998). *Codes and Cryptography*. Clarendon Press.

Wiedemann, D. (1987). Quantum cryptography. *Sigact News 18*(2), 48—51.

Wiesner, S. (1983 (original manuscript 1969)). Conjugate coding. *Sigact News 15*, 78—88.

Williams, C. P. and S. H. Clearwater (1998). *Explorations in quantum computing*. TELOS.

Williams, C. P. and S. H. Clearwater (2000). *Ultimate Zero and One: Computing at the Quantum Frontier*. Springer–Verlag.

Wolf, S. (1999). Unconditional security in cryptography. In *Modern Cryptography in Theory and Practice, Proceedings of 1998 Summer School in Cryptology and Data Security*. Springer–Verlag. Volume 1561 of *Lecture Notes in Computer Science*.

Wootters, W. K. and W. H. Zurek (1982). A single quantum cannot be cloned. *Nature 299*, 802—803.

Wyner, A. (1975). The wire–tap channel. *The Bell System Technical Journal 54*(8), 1355—1385.

Yamamoto, Y. (2004). Physics of quantum information: Lecture notes. NII Quantum Information Science Group.