Master Thesis ELE/FMI/1998-209

Protocols in Quantum Cryptography Systems:

Implementation of Software for Secret Key Extraction

by

Daniel Ljunggren

A Master of Science Thesis in
Electrical Engineering

Supervisor: Ph.Lic. Mohamed Bourennane, KTH
Supervisor and Examiner: Associate Prof. Anders Karlsson, KTH

Stockholm, February, 1999

Protocols in Quantum Cryptography Systems:

Implementation of Software for Secret Key Extraction

by

Daniel Ljunggren

A Master of Science Thesis in
Electrical Engineering
(M.Sc.E.E.)

Supervisor:
Ph.Lic. Mohamed Bourennane, KTH
Supervisor and Examiner:
Associate Prof. Anders Karlsson, KTH

Laboratory of Photonics and Microwave Engineering
Department of Electronics
Royal Institute of Technology (KTH)
Stockholm, Sweden, 1999

Abstract

Protocols in Quantum Cryptography Systems: Implementation of Software for Secret Key Extraction

by
Daniel Ljunggren

The relatively new technology of *quantum cryptography* is currently subject to vast development. During the last decade, several groups have reported many good and promising results of experimental work. Further research will contribute to deeper knowledge within the wider research-field of *quantum information*.

Quantum cryptography is the art of distributing secure keys between two parties by modulating digital bits into photon-states, so-called *qubits*. These photons are sent over the *private quantum channel*, i.e. an optical fiber. Particles like photons rule under the laws of quantum physics, such as the *complementary principle* and the *no cloning theorem*. As an effect, any eavesdropping-attempt can be discovered.

Classical cryptography systems rely upon the supposed difficulty of solving certain classes of mathematical problems. Quantum cryptography does not have those limitations - only physical laws pledge for security.

The work performed at KTH has reached the stage of developing computer-controlling software in order to automatically transmit a random bit-string, the so-called *raw-key*. Additional classical communication is necessary to extract a *secret key* from the raw-key. This communication is held over the *public channel* and it requires the use of *protocols*. These protocols include software that can govern all public discussion, and algorithms that bring out a perfectly secret key.

This Master's Thesis report will try to explain the underlying methods of these communication protocols and inform about the work done implementing this software. Several methods of *reconciliation* (error correction) are investigated. Eventually secret information is leaked during the key distribution process, however, this information can be ignored by utilizing the method of *privacy amplification*, where the key is compressed by hash-functions into a shorter key with an arbitrarily high security level.

The software is capable of simulating the whole quantum channel, and thus the complete key distribution process including protocols. The best performing protocols are implemented. We investigate the effective *bit-rates* achievable. Other simulation results are used to draw conclusions about implementation practicalities.

Acknowledgements

Believe it or not but you are now confronting a copy of my diploma-work required for the M.S.E.E degree. First, I'd like to appreciate myself for making such a good choice in the selection of a thesis-proposal as it turned out to be. I had a real good time. This in spite of some real late hours necessary in order for this final report to become ready. I suppose I must've been the first student who ran short with time!?

This work would have been neither possible, nor ended with good results without my supporting supervisor and examiner *Anders Karlsson*. Thanks. Special thanks also to my supervisor *Mohamed Bourennane* who's been kind to explain many things, so both to confuse and to clarify, but luckily mostly the latter. Appreciations also to *Norbert Lutkenhaus* at the University of Helsinki, Finland, for helpful discussions and valuable feedback, especially concerning background theory and formulas of section 6.3.3. Thanks *Gunnar Bjork*, head of Quantum Optics group, for having me investigate coding.

Also, thanks to all Ph.D. students at the department for the numerous floor-hockey disputes. I had two roommates the first half of my time here; thanks *Maria* and *Ulf* for the company.

The person who suffered most from my late hours was with no doubt my sweet *Sara*. I think she endured finally.

Daniel

A stormy winter day in February the year of 1999

Table of Contents

Project and Paper Organization

This paper completes a diploma-work submitted in partial fulfillment for the requirements of the academic degree: Master of Science in Electrical Engineering. The project was conducted in the Quantum Optics* group at the Laboratory of Photonics, Royal Institute of Technology, in an ongoing research project towards a realization of a real Quantum Cryptography System. Within this project a partial goal was to build up a software interface for the experimental quantum cryptography system build in the lab. The software should be able to simulate the QC-system and eavesdropping attempts, as well as evaluating the performance of the implemented communication protocols. A first step was to become acquainted with the rich topic of quantum cryptography by literature search and reading. Thereafter, concrete work was set out in order to reach the following milestones set up in anticipation:

- Make software tools for extraction of a secure cryptographic key, i.e. MATLAB-code.
- Implement MATLAB-code for error-correction and privacy-amplification in a real cryptographic system.
- Simulate a quantum cryptography system in the presence of eavesdropping and evaluate performance.
- Evaluate the software in order to illustrate the practical aspects of a real implementation.

The first chapter gives an introduction and motivation for Quantum Information, also describing previous work and future possibilities. Thereafter, in chapter 2, Classical Cryptography is treated in order to relate this to the area of Quantum Cryptography (QC), which is the subject of chapter 3. Here is explained the main principles behind QC and different protocols like the BB84 and B92. In Chapter 4 is presented the method of obtaining a perfectly secure key between two parties sharing common information that is only partly secret. Different reconciliation methods are presented, the most important: *cascade*. Also investigated is the method of *privacy amplification*. Chapter 5 presents experimental work performed to this date (in particular for the laboratory being the seat of undersigned). The interface and software implementation is described in chapter 6 together with simulations evaluating the performance of the protocols. The paper is closed with conclusions in chapter 7.

Happy reading!

© *Daniel Ljunggren*, *Stockholm*, *February*, 1999

* The group shortly thereafter formed their own laboratory: Quantum Electronics and Quantum Optics at the same department.

Chapter 1

Quantum Information

1.1 Introduction

A veteran in the field, Artur Ekert once said, "*there is potential here for truly revolutionary innovations*." He was talking about the rapidly increasing area of Quantum Information. In 1935 Einstein, Podolsky and Rosen proposed the famous EPR paradox that now symbolizes the mysteries in quantum mechanics. At the same time, the first brick in the foundation for today's computers was laid; namely the Turing machine, invented by Alan Turing. The computer industry has become the largest business of the world today.

Although knowledge in quantum physics is necessary for understanding of the solid state technology in the computers, the information process has remained a classical process. Will a change eventually come true? The use of quantum physics could revolutionize the way we communicate and process information. The important new observation is that information is not independent of the physical laws used to store and process it. The new approach is to treat information as having those quantum properties it is really equipped with. What new insights can be gained by encoding information in individual quantum systems? In other words, what happens when transmission and processing of information are governed by quantum effects without classical counterparts?

For example, while classical bits must take on one of the two mutually exclusive values of 1 and 0, then superposition of quantum states enables a bit of information

to have the strange property of "being" 1 and 0 simultaneously. This may be seen as a threat or an opportunity. The quantum information processes may have a long way to go before they can rival the classical counterparts, nevertheless they have shown potentials already.

Quantum cryptography, the topic of this paper, is the most mature area of quantum information and has been very successful in experiments. The issue for this application is no longer whether quantum cryptography works, rather if it can be made robust and practical enough for commercial use. Research is performed both in academia and in companies. For further introductory reading, see [36].

1.2 The Computer Challenge

The idea that nature can be controlled and manipulated at quantum level is a powerful motive for physicists and engineers. The area of quantum information is truly interdisciplinary and gathers researchers from very separate fields of interests.

Maybe the most intriguing challenge in quantum information is to develop a computer in possession of enormous computing power. Progress in this direction is taking place almost every day, although these computers today are limited to individual gates and flip-flops-states.

Quantum mechanics, far from putting limits on the classical computations, will provide completely new modes for computation, including algorithms that can perform tasks at most in polynomial time. This important and remarkable feat has been proven for quantum computers.

So, if quantum computers become true it this means that all of today's cryptography algorithms will fail to provide secure communication, as these are based on the fact that today's computers is limited in computational performance. Maybe it will even be enough with the exponentially increasing power of today's classical computers to break the algorithms in future. Quantum cryptography is not vulnerable to this threat. Further reading, [35].

Chapter 2

Classical Cryptography

2.1 Introduction

Cryptography is the art of encrypting plaintext into secret messages. The rival to cryptography is cryptanalysis, the art of breaking ciphertexts. The science of sending secret messages and keeping them secret to undesired spectators is combined by these two branches into what's called Cryptology. The word Cryptology origins from ancient Greek. It means "hidden word". The history of this science is as long as the written word it self, and can be found in many textbooks [29].

Until not long ago, cryptography was mostly a concern for the military. During the past centuries there has been a considerable interest for further developing of secret communication. However, the race towards unbreakable cryptography started many years before our time, trails can be found to the Romans over 2500 years ago. They wrote down secret messages on papers winded around cylinders with a certain diameter. Only a person possessing a cylinder with the right diameter could then read the message.

However, for the past decades there has been a dramatic change. Going from being military business, cryptography has, with today's computerization developed into a civil matter as well. Anyone ever tried to use cash-cards, internet shopping, distance-banking, etc. knows the importance for everyday people to have access to secure communication.

Encryption is the process of taking a plaintext message and scrambling it so that it becomes unreadable to anyone, except the authorized receiver who has a key to decrypt it. Encryption produces a ciphertext. The process of turning a ciphertext back into a plaintext is called decryption. The reason for the encryption into a ciphertext is just to make sure that an eavesdropper cannot read the message. See Figure 2.1.

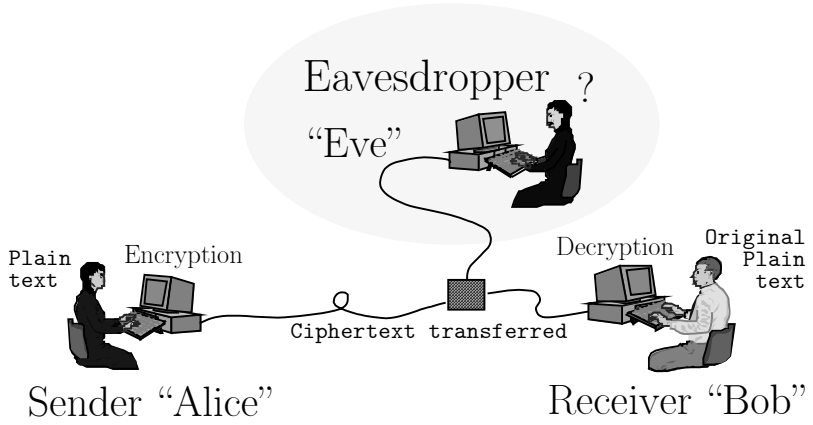


Figure 2.1: The basic problem of distributing a key between Alice and Bob is to make a safe transfer despite eavesdropping attempts. A plaintext message is encrypt using the key, into a ciphertext. Only a person with the same key can decrypt it back to a plaintext message.

In order to make the terminology and discussions in cryptography more clear, we have several players acting on the field; Alice is the sender of the information, Bob is the authorized receiver, and the malicious eavesdropper is of course named Eve.

There are several cryptosystems, all of them based on the same idea: Alice and Bob share a key that can lock and unlock a message. This key has to be distributed to authorized individuals in some way. Once you have a key there is no problem to encrypt a message. This is the "key" problem in cryptography; how to safely distribute a key between two parties. Today, different high-technology protocols can accomplish this task. In ancient times this was of course most easily done through a courier, trusted by both parties, and maybe still today this is the most secret way, although very unpractical.

We shall first briefly present two classes of cryptosystems comprehending the most popular protocols of today. The secret-key cryptosystem uses *symmetric keys*; the same key is used for encryption and decryption. The public-key cryptosystem includes the very popular RSA-algorithm, and uses an *asymmetric key*, i.e. different keys are used for encryption and decryption.

2.2 Secret-key Cryptosystems

Secret-key cryptosystems, also called symmetric algorithms, are algorithms where the encryption key can be calculated easy from the decryption key and vice versa. They are often the same. These mathematical algorithms require that the sender and receiver agree on a key before they can communicate securely. The security relies on

the secrecy of the key, if anyone get hold of the key he could also read the ciphertext messages. The key must remain secret in order to have secret communication. In mathematical notation the encryption and decryption can be denoted:

$$e_k(m) = c$$
$$d_k(c) = m$$

where e_k and d_k is the encryption-key and the decryption-key respectively.
The Data Encryption Standard (DES) is an example of a secret-key cryptographic protocol. The algorithm is a block cipher that encrypts 64-bit size blocks of a message. Although quite old, this algorithms still today provides one of the highest levels of security.

2.3 Public-key Cryptosystems

Public-key cryptosystems is an asymmetric key algorithm were the key used for encryption is different from the key used for decryption. Furthermore, the decryption key cannot, within a reasonable amount of time (with today's computers) be calculated from the encryption key. The idea is to make the encryption key *public*. Anyone can then encrypt a message using this public encryption key, but only a specific person can decrypt the message. It is this person who supplies the encryption key for encryption of messages addressed to him. The best known and most popular public-key cryptography protocol is the RSA system.

Encryption and decryption is denoted:

$$e_n(m) = c$$
$$d_n(c) = m$$

The two keys are generated using the Euclidean Algorithm and the Chinese Remainder Theorem, well known algorithms in mathematics [30]. The idea is to choose two large prime numbers p and q. Knowing only the product $n{=}pq$ of these numbers it is hard to solve for the two factors. The encryption-key, e, is chosen randomly. The decryption-key d is calculated using p, q and e. Let e and n become public. These then make up the public key and with them, you can encrypt a message. However, only the person knowing d can decrypt the message. The private key, d, cannot (easily) be calculated from e and n.

2.4 Key Distribution and Protocols

In order to distribute keys we need to define *protocols*. A protocol is a series of steps, involving two parties or more, designed to accomplish a specific task. Protocols can for example use the algorithms described above in order to distribute keys. For the case of quantum cryptography, a key distribution start with the distribution of a raw key over a *secret channel*. See Figure 2.2. Despite the name of the channel, this string of bits might not be fully secret. Alice and Bob have to agree in some way to distill - from the raw key - a secret key shared only between the two parties. This agreement has to be in form of public communication between Alice and Bob performed over a *public channel*. Any information sent here will become known to the eavesdropper. The task of the protocol is to make up a frame for this communication.

Everyone involved in a protocol must know the protocol and all of the steps to follow in advance. Note that Eve will certainly know the protocol. For a cryptographic protocol can be said, it should not be possible to do more or to learn more than what is specified in the protocol. An eavesdropper can certainly try to attack an protocol and learn some information from it. It is important for Alice and Bob being able to estimate the amount of this information. Once a secret key is distributed, secure communication can be guaranteed.

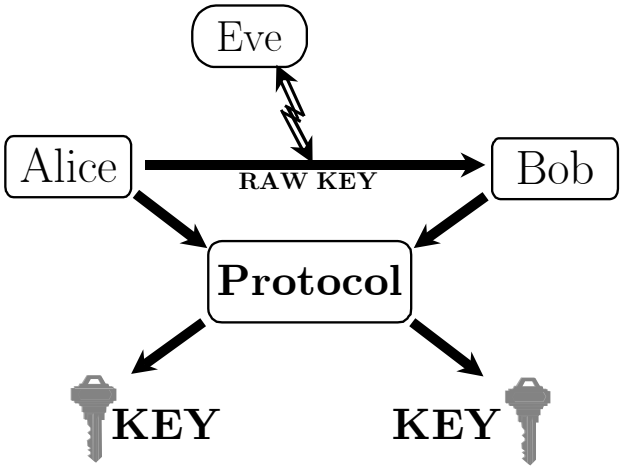


Figure 2.2: A protocol is a collection of definitions, setting the bounds of communication allowed between the two parties, in order to agree on two identical shared keys. Everyone involved in a protocol must know the protocol and all of the steps to follow in advance.

2.5 Security in Classical Cryptography

Full security can be achieved between two legitimate users if they share a completely secret key. By definition, a key is considered completely secret if an eavesdropper has no other method in finding the key than making a completely random guess for each and every bit making up the key.

Interestingly, the only cipher that cryptologists have ever proved to be secure is the One-Time Pad (OTP). In the OTP the key is as long as the message itself and the cipher is attained by computing the XOR of each bit of the message by the respective bit of the key. In this way a 4117 bits key is needed to encrypt a 4117 bits long message. The key must be truly random, i.e. not generated by a pseudo-random generator; (section 2.6) otherwise, the cipher may be crackable. OTP:s are of course a bit impractical and therefore rarely used: we need a key as long as the message itself. The key is only to be known by the sender and the receiver and must be delivered over a secure channel. The key may also be used only once, or an attacker could succeed to break the message.

The weak link for classical cryptography is that it relies to the speed, or rather to the low computational power of today's computers. All algorithms in classical cryptography are hard to break because they need considerable amount of computing power and time to be cracked. For the future, and especially if very fast quantum

computers become true then classical cryptography may be doomed. If, or should you say when, this happens, then quantum cryptography may be the only solution.

2.6 Random Bit Generators

The generation of keys is definitely the most sensitive operation in cryptology. Only in a system where the key is build by numbers that are truly random it is possible to reach security, i.e. a key that is completely unpredictable by an attacker. To get these random numbers we can not use the so-called pseudo-random generators used by computers. Truly random numbers are different from these computer-deterministic numbers mostly used for simulation purposes and games. Pseudo-random numbers are generated by algorithms that only look as if they were random. They are actually deterministic numbers from a sequence repeating itself with a certain period - not good for cryptography. Truly random numbers can be derived only from the environmental *noise* of the physics world, such as the thermoelectrical noise over a diode.

Therefore, in the implemented Matlab program the random bit-generator `randn()` will used only for small simulations. For real key transmissions and large simulations we have to use an external noise-generator from where it is possible to get truly random bits. This generator is a device connected to the serial port of the PC. We use the random number generator from the manufacturer Protego. C-files are provided to handle the communication with the serial port. Preparations are made in the program in the prospect of future implementations and use of the device. This is concerned with in Chapter 6.

Chapter 3

Quantum Cryptography

3.1 Introduction

The past century physicists have discovered we live in a quantum world. Everything surrounding us, and influencing our lives acts by the laws of quantum mechanics. Still though, most of mankind's communication is based on classical physics and information processing. Already realized is that quantum physics is more than a radical sidetrack of classical physics. Many new possibilities are offered for information processing.

Quantum cryptography is among the first real-world applications to use the theory of quantum physics. It has for long now entered the mature age. Quantum cryptography is based on the postulate of quantum physical that says, *every measurement perturbs a system*, called Heisenbergs uncertainty. For example, think of sending photons, a particle especially suited for enlightening the properties of quantum mechanics, coded with digital 1:s and 0:s in different polarization states. If the receiver in some way can tell if these photons have been disturbed during transmission, he can also tell weather someone tried to eavesdrop or not. This is possible in quantum cryptography. Several schemes of communication protocols have been proposed, the first in the mid-80:s. Quantum cryptography uses the seemingly limiting but potential property - every measurement perturbs a system - and turns this into a useful process where any eavesdropper is forced to reveal her presence. Thus, Heisenbergs postulate has given us some positive use in the art of secure and

private communication. Mathematicians have since long imperfect solutions to the problem of security. Albeit, a system is never more secure than its weakest link and the mathematical solutions are based on hard solved algorithms, but only hard in the sense of today's computing power. With the computing power increasing at remarkable pace, and considering the possibility of future quantum computers, classical cryptography may have one day meet its match. Quantum cryptography does not have that limitation, an eavesdropper may have access to unlimited computing power, she will still be limited by the laws of physics.

The first key distribution scheme for quantum cryptography was published in 1984. The protocol is best known in the quantum cryptography community as the BB84 protocol. It was proposed by Bennett and Brassard [4]. This may be the first time quantum effects of nature have been directly exploited to give a fundamental advantage in information processing. In 1992 another protocol known as B92 was proposed [2]. This scheme uses a different property (phase rather than polarization) of the photon encoding the bits, but is otherwise based on the same principles as BB84. The two protocols will be explained in this chapter. Other schemes have also been proposed that use entangled states, two orthogonal states, and six states.

So, how does quantum cryptography work? Well, the complete answer to this question will take the rest of this chapter and the next to explain. However, we will now first describe the underlying procedures briefly.

Individual photons, or *light quanta*, are encoded in orthogonal quantum states to carry information about either 1:s or 0:s and send in either one of several non-orthogonal bases. These bits make up the cryptographic key. The photons can for example be encoded in states of polarization or phase. The principle of Heisenbergs uncertainty now ensures the security of the transmission, but not all by it self as we will understand later.

Random bits at Alice are encoded onto photons according to the protocols mentioned above. Usually an optical fiber is the transmission-medium for these photons. The laser-generated photons are then send through the fiber, called the Quantum Channel, where Eve might try to eavesdrop some photons. See Figure 3.1. These photons are then received and decoded by the detector at Bob. According to the protocol, Alice and Bob then agree to use only a fraction of the bits send originally by Alice. This procedure is the discussion of signal states over the Public Channel, called *sifting*. Alice and Bob now each share a version of the bit-string, albeit they are not exactly the same as errors may be present. Then *error correction* is applied, where Eve will learn additional information about the key. To reduce the eavesdropper's information a technique of *privacy amplification* is applied where the key is compressed. After this final step, Alice and Bob will share a completely secret key, on which Eve as an arbitrarily small amount of information. Once this secret key is established it can be used together with classical symmetric key cryptographic techniques to let the two parties communicate secretly. Further introductory reading e.g. [36],[35], advanced e.g. [3],[10],[16],[26].

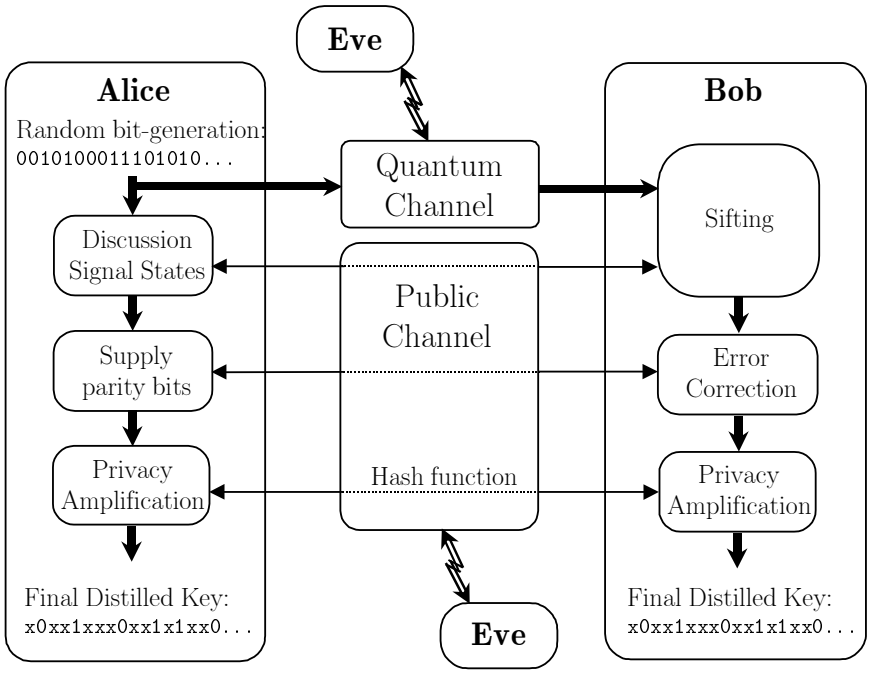


Figure 3.1: Diagram of the basic flow in the information process for quantum cryptography. Two channels; the Private Quantum Channel and the Public Channel are shown. Several steps like Sifting, Error correction and Privacy Amplification are needed to distribute a final secure key between the two parties, Alice and Bob. An eavesdropper Eve is maliciously persisting on both channels.

3.2 Quantum Physics Interpretation of Coding

So, how about this coding of the bits into quantum states? We will best understand this by first briefly stating the properties of quantum states. See e.g. [21].

A photon can for example be encoded in either polarization or phase. These optical properties are examples of quantum states of a photon. Let a quantum state be denoted by $|\psi\rangle$, and let us discuss the case of polarization. Two polarization states $|\psi\rangle$ and $|\phi\rangle$ are orthogonal if their inner product $\langle\psi|\phi\rangle$ is equal to zero. A basic property is that two quantum states can be in superposition, that is $|\psi_s\rangle = a_0|\psi\rangle + a_1|\phi\rangle$. Quantum states rules under the Schrodinger equation,

$$i\hbar\frac{\partial}{\partial t}|\psi_s\rangle = \hat{H}|\psi_s\rangle$$

which is linear. Therefore, if both $|\psi\rangle$ and $|\phi\rangle$ are solutions to this equation, then $a_0|\psi\rangle + a_1|\phi\rangle$ is also a solution for all other choices of a_0 and a_1. If we try to measure (learn) the state (a_0 and a_1) of $|\psi_s\rangle$ then quantum mechanical laws, the complementary principle, says that we will perturb its state. A measurement is said to be indeterministic. In a photo-detector, the photon-state will vanish as the photon is absorbed by the detector. Instead, the probability of finding $|\psi_s\rangle$ in the state $|\phi\rangle$ is given by $|a_1|^2$, and in state $|\psi\rangle$ by $|a_0|^2$. We cannot for sure tell in what state an individual photon sent to us is without measuring it and thus collapsing its state.

Also known, is that no quantum state can be cloned perfectly. These properties build up the main foundations whereupon quantum cryptography rests.

If two superposed states are orthogonal to each other, i.e. $\langle\psi_s|\phi_s\rangle$=0, then one case implying this, is when one of the coefficients a_0 or a_1 is zero and the other is equal to one, (because $|a_0|^2+|a_1|^2$=1). In this case, trying to measure a state $|\psi_s\rangle$ to se if it is in state $|\psi\rangle$, the result will with *unit* probability give that it was in the state of either $|\psi\rangle$ or $|\phi\rangle$, depending on what state it was in, and vice versa. On the other hand if two states are non-orthogonal to each other, lets say with a 45° difference ($|\phi_s\rangle$=$|\phi\rangle$ and $|\psi_s\rangle$=$1/\sqrt{2}\,|\psi\rangle+1/\sqrt{2}\,|\phi\rangle$), then measuring $|\psi_s\rangle$ to see if it was in state $|\psi\rangle$ will give with a probability of $|a_0|^2$=$|a_1|^2$=1/2 that it was in either state of $|\psi\rangle$ or $|\phi\rangle$. This is a random guess. A conclusion can be made, for a measurement to give a deterministic result you have to do a correct assumption of what state to test against. This state has to be orthogonal to or parallel with the state to be measured. Quantum cryptography use these properties of orthogonal and non-orthogonal states to code bits and to guarantee security.

For convenience, or maybe as an effort in making quantum information even more fun (?), physicists developing quantum information has coined the term *qubit*= "quantum"+"bit". By that is meant just what it sounds like; information in form of a digital *bit* represented by a *quantum* state of an atom, ion or photon. In the same way as a bit is a unit of information in classical information, a qubit is a basic unit of information in a quantum information system. In corresponding notation, we have for the qubits $|0\rangle$ or $|1\rangle$, i.e. bits 0 and 1 coded in quantum states $|\psi\rangle$ and $|\phi\rangle$.

3.2.1 Non-Orthogonal Bases

Alice now codes her random bits into orthogonal states as $|0\rangle$ or $|1\rangle$, but in addition, she sends them randomly in either one of two non-orthogonal bases. A base consists of two orthogonal states (90° angle) and is rotated 45° compared two the other two orthogonal states of the other base. See Figure 3.2. This means that a photon will be send in either one of all four polarization-angles, and is therefore called 4-*state scheme*.

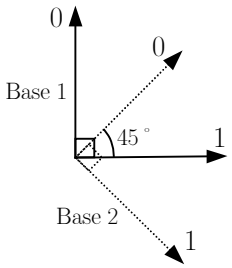


Figure 3.2: Two non-orthogonal bases are used and chosen randomly by Alice when encoding her bits. The bits 0 and 1 are represented by orthogonal states in either one of the bases as shown in the figure.

Remembering from the previous discussion that measurements has to be performed using an orthogonal state, i.e. the same base in order to be deterministic, this implies that Bob has to measure in the same base in order to be sure about the measurement outcome. That is, with unit probability tell correctly if either $|0\rangle$ or $|1\rangle$ was sent by Alice. If he uses the wrong base he will receive either $|0\rangle$ or $|1\rangle$ with 50% probability

and this bit is then not useful. Bob will detect and demodulate all incoming photons using a random choice of base. So, 50% of the time he will receive the correct bit. After transmission of all bits, Bob will publicly announce to Alice what bases he used, and Alice will give information back of what bits where measured in correct base. They keep only those bits with bases matching. But, Bob will never tell what result he got, and if there is no eavesdropping and small noise Alice and Bob will now share a common, random bit-string.

The artifice using this complicated procedure is that an eavesdropper can be detected. Eve will face the same problem as Bob, not knowing what base to measure in. If she uses the *intercept and resend* strategy (treated in detail later) for eavesdropping she will introduce 25% of errors. The essence of this strategy is for Eve to cut the fiber, detect all photons, and resend them to Bob. But, in half of the cases she have will resent in the wrong base, and so Bob will use the wrong base on additionally 50% of the photons. Therefore 25% errors are introduced between Alice's and Bob's version of the key, and Eve can be detected easily.

3.2.2 Polarized Coded Information

We continue discussing polarization states, but will now define how to encode. For example, we can choose four different states, $|\leftrightarrow\rangle$, $|\updownarrow\rangle$, $|\circlearrowleft\rangle$, $|\circlearrowright\rangle$, where, $\leftrightarrow$ is horizontal polarization, $\updownarrow$ vertical, $\circlearrowleft$ left-circular, and $\circlearrowright$ right-circular. $\leftrightarrow$ and $\updownarrow$ belongs to the linear polarized base, $+$, and $\circlearrowleft$ and $\circlearrowright$ belong to the circular base, $\circlearrowleft\circlearrowright$. These two bases are non-orthogonal, actually with 45° difference. Let us define that $|\leftrightarrow\rangle$=0, $|\updownarrow\rangle$=1, $|\circlearrowleft\rangle$=0, and $|\circlearrowright\rangle$=1. This definition is agreed between Alice and Bob. Alice then chooses from one of these four states to encode her 1:s or 0:s, and the task for Bob is to decode this information. If the qubits were sent in left- or right-circular polarization he received the correct bit only when he used the circular base for measuring, and for qubits sent in vertical or horizontal polarization he will have received the bit only if he used rectangular polarization.

3.2.3 Phase Coded Information

The qubits can also be encoded in the phase. For a 4-state scheme, like discussed above we need four different phases, being coded by 1:s and 0:s. Let us define the following two non-orthogonal bases: rectangular $+$ and diagonal $\times$. We can use the angles, $|\rightarrow\rangle$=0°, $|\uparrow\rangle$=90°, $|\nwarrow\rangle$=135°, and $|\nearrow\rangle$=45°. The bits are encoded each in two orthogonal states, $|\rightarrow\rangle$=1, and $|\uparrow\rangle$=0, respective $|\nwarrow\rangle$=1, and $|\nearrow\rangle$=0. See Figure 3.3. The principle is the same here, Alice chooses randomly one of the two bases to encode her bits. Bob randomizes his measurement in either rectangular or diagonal base and will end up with 50% of the bits correct. Bob has to measure in rectangular polarization in order to receive the correct bit for bits send in 0° and 90°, and for 135° and 45° the correct bit is received for Bob if measured in diagonal polarization.

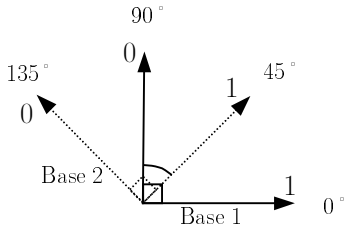


Figure 3.3: Phase encoding using two non-orthogonal bases. The bits 0 and 1 are represented by orthogonal states in either one of the bases as shown in the figure.

3.3 Quantum Key Distribution Protocols

Several schemes or protocols have been proposed that use the properties of quantum mechanics to encode information. The most common methods use the polarization, phase, or phase difference. We will describe how a practical protocol works. The ideas behind are already presented. The steps following the so-called *raw bits* transmission over the quantum channel were seen already in Figure 3.1. These steps, reconciliation and privacy amplification are also part of a complete key distribution protocol. Here though, we shall restrict ourselves to the schemes that are part of the so-called $sifting$ procedure.

3.3.1 The BB84 Scheme

The BB84 protocol was the first protocol to be invented [4], and its based on encoding of the qubits by use of polarization. In this example of a key distribution the 4-state scheme is used as coding where Alice chooses between four possible states in two bases. For a basic quantum key distribution protocol, see Table 3.1. It begins when Alice starts to set the polarization of the photons, choosing from the four different states, and sends them to Bob. [15].

1: Alice sends a random sequence of photons with the choice of either polarization base rectangular or circular, where, ↔=0, ↕=1, ↺=0, and ↻=1. Alice records the sent bits and the time they were sent.

2: Bob measures the photons polarization in a random sequence of the two bases. Bob records the detection outcome and time when he received the pulses.

3: Alice and Bob then publicly compare their bases used for every bit sent and keep only the data from the measurements where they used the same base. Similarly they agree to discard bits where Bob's detector failed to detect any photon, a fairly common event with existing detectors.

4: The resulting binary key should then be the same for Alice and Bob. Eventually errors has though been introduced since we have no perfect detectors and eavesdropping might have taken place.

5: Alice and Bob now perform error correction. If there are too many errors, the error correction method will not be able to correct all errors. But even if error correction performed fine, to many errors implies that the key will be compressed

asymptotically into a zero-length key, and thus in that case Eve has revealed her self by introducing too many errors. No key is then distilled.

6: Otherwise they now share a completely error free key.

1. Alice	Base:	+	↻	↻	+	↻	+	+	+	↻	↻	+	↻	+	↻	↻
	Encoding:	↔	↺	↺	↕	↺	↔	↕	↔	↺	↺	↔	↺	↕	↺	↺
	Bit:	1	1	0	0	0	1	0	1	1	1	1	0	0	1	1
2. Bob	Base:	+	↻	+	+	+	+	↻	+	↻	+	↻	↻	↻	↻	+
	Received bit:	1	-	1	0	0	0	-	1	1	0	1	0	-	1	0
3. A&B	Same base:	✔	✔		✔		✔		✔	✔			✔		✔	
4. Alice	Kept bits	1			0		1		1	1			0		1	
Bob	Kept bits	1			0		0		1	1			0		1	
5. A&B	Error correction						✔									
6. A&B	Error free key:	1			0		1		1	1			0		1	

Table 3.1: Example of a quantum key distribution protocol. There are two non-orthogonal polarization states; ↻ (circular) and + (linear). Failure in detection of a photon is marked -.

3.3.2 The B92 Scheme

The B92 scheme was proposed by Bennet [2] (guess when?) in 1992. It uses a slight modification, or maybe one should say a simpler encoding, with only two states. It is therefore called the 2-*state scheme*. In this scheme Alice chooses randomly between only two non-orthogonal states and sends the encoded photon to Bob. See Figure 3.4. As these are non-orthogonal, there is no way for Bob or Eve to do deterministic decoding. Still, a measurement can be performed that will half of the time fail to give an result and the other half give the correct result. The angles are coded as, $|\uparrow\rangle$=1,and $|\nwarrow\rangle$=0, where $|\uparrow\rangle$=90°, and $|\nwarrow\rangle$=135°. Any other two non-orthogonal states would work. Bob uses ✕=Base2 to detect if there was sent a $|\nwarrow\rangle$=0 state as this is orthogonal to the base. By the same argument he uses +=Base1 to detect if the state $|\uparrow\rangle$=1 was sent. If he uses the wrong base he will get an inconclusive result ?. By this is meant that he does not detect any photon at all. If for example Alice sends to Bob a 0, then he might get either a 0 or an inconclusive result ?, but never a 1. A photon will never trigger Bob's detector if he uses the wrong base, but will trigger if they use the same base. Bob announces over the public channel if he detected a photon at all or if he did not. Only outcomes from measurements where Bob received a photon is kept, all other corresponding bit-positions of the sent bit-sequence are simply ignored. They will arrive at a commonly shared key. The security of this scheme relies on the same properties as for the BB84 protocol, i.e. the

problem that even Eve does not know in what bases to measure, and therefore she introduces errors.

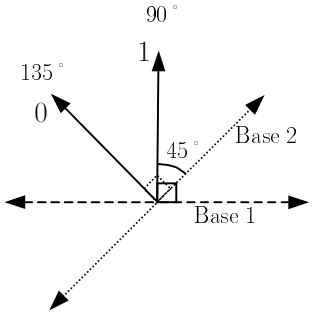


Figure 3.4: BB92 scheme. Phase encoding using two non-orthogonal bases. The bits 0 and 1 are represented by non-orthogonal states and are in turn orthogonal to either one of the measuring-bases as shown in the figure.

3.3.3 Other Schemes

There are numerous ways to implement qubit-encoding in practice. One way for easy realizations made for an experiment is to use phase encoding with interference. That is, we detect phase differences by interference between photons. This is easy to detect and therefore used in experiments. This will be described in further detail in Chapter 5.

Another scheme is the 4+2 protocol, which combines the ideas of BB84 and B92. As in BB94 Alice chooses between two different bases (such that there are 4 different states) and as in B92 the two states within a basis are non-orthogonal. This gives a total of six states. Another protocol, the *six-state* scheme, uses both circular, rectangular, and diagonal polarization states to obtain a total of six states.

Finally another one, is the *Ekert* scheme, [12] that uses entangled states to encode the qubits. Bell's inequality in quantum physics is used to find out if any eavesdropper might have been interfering with the transferred data.

3.4 Eavesdropping Models

The security of quantum cryptography relies on the fact that different non-orthogonal states cannot be distinguished with accuracy. Eve never knows what base to use for detection of the photons. An eavesdropping attempt will therefore introduce errors in the transmission that can be detected. The key problem for Alice and Bob is to find out to what degree Eve has information on Alice's and Bob's shared bits. How much can she gain by eavesdropping the channel, and what methods does she have at disposal? To get an estimate on this quantity is very important, as we will se later the security gain (by compression of the bits in privacy amplification) relies on this amount. We will denote this amount by t, and it is defined as the fraction of Bob's sifted bits that Eve has learned by eavesdropping the quantum channel. There are no general methods to estimate this amount and it has to be assumed that Eve has every method imaginable available. However, we will here restrain this argument to only concern attacks by Eve on signals (transmitted

qubits) individually. *Coherent* and *Collective* attacks are briefly stated in the concluding section.

It is hard to exactly estimate the knowledge that Eve might have harvest. However, we can calculate for different upper bounds on t. In the following two section we'll explain two different classical and basic approaches of eavesdropping, namely, *intercept/resend* and *beamsplit*. We will derive formulas for t. However, these estimations are straightforward and simple. There are estimations of t performed more rigorously with considerations taken to many more possible ways of attacks. See paper by Lütkenhaus, [17]. The goal is to get an upper bound on the expected average information Eve achieved on the error corrected key. The fraction t is a function of the errors-rate her eavesdropping strategies persuade to the signal. Two bound are given in this section, one for were the errors are to be discarded in the error correction protocol and the other for corrected errors. The fraction (w.r.t. N_s=nr. of sifted bits) of bits learned for both versions of error correction are (no leaked information about the position of errors):

$$t_{crude} \le \begin{cases} \log(1+4e-4e^2) & \text{for } e \le 1/2 \\ 1 & \text{for } 1/2 \le e \end{cases}$$

In the remaining part of this paper this bound will be referred to as the *crude* bound.

We will now instead turn to what I refer to as the *simple* bound. This is discussed in [3] by Bennet et al. The sum of information leaked for both intercept/resend (u) and beamsplit (v) is given by the fraction

$$t_{simple} = u + v \le \mu + 4e/\sqrt{2} + 5\sqrt{(\mu(1-\mu)+(4+2\sqrt{2})e)/N_s}\ .$$

In order to distill a final key with length≥0 we have for the simple estimate a theoretical maximum allowed error-rate of 15%. For the crude estimate the bound is 11.5% (these limits apply for error correction at Shannon limit, treated in Chapter 4). This is why the latter is called the crude estimate.

In the following derivations are assumed that Eve has unlimited technology available, but consistent with quantum mechanics. She can also store light-pulses for an arbitrarily long time before measuring them.

In the proceeding sections, we have assumed that we can generate single photons and demodulate them each with the information of a bit, 1 or 0. However, this is not really the case, since it is difficult to generate single photons. Therefore, existing schemes relies in weak pulses instead. These weak pulses will have an intensity of typically μ=0.1 photons per transmission. The probability that two or more photons are send will then be given by $\mu^2/2$ (a laser exhibits a Poissonian distribution of μ). It is important that this information about one bit is transported by only one single photon. We will see that due to this, the photon transmission is sensitive to the beamsplit attack.

3.4.1 Intercept/Resend

Assume that Eve has a perfectly efficient detector. In Intercept/resend, Eve intercepts selected lightpulses sent through the fiber and reads them in bases randomly chosen by her. See Figure 3.5. For each pulse, with probability approximately μ Eve's perfectly efficient detectors will succeed in detecting a photon. When this occurs Eve resend a new photon modulated in the same way. To avoid suspicion Eve's resent pulses should be of such intensity as if no eavesdropping had taken place, i.e. yield the same rate of pulse detection at Bob as she had.

But, as shown in previous section at least 25% of the pulses resent by Eve will be modulated in the wrong base and give a error result for Bob. Moreover, each of these intercept/resend bits are not worth more to Eve than if she where told Alice's bits with probability $1/\sqrt{2}$. From the paper by Bennet, [3], can be learned that Alice and Bob now conservatively estimate that fewer than a fraction of $4e+5\sqrt{12}e$ bits have been subjected to this eavesdropping strategy, where e is the error-rate picked up by Bob. The second term is an arbitrarily standard deviation. Alice and Bob can now draw the conclusion than Eve cannot have learned more than

$$u_{intercept/resend} = 4e/\sqrt{2} + 5\sqrt{(4+2\sqrt{2})e/N_s}$$

bits of the key. Of course, in a practical realization, errors can arrive from other natural causes, as detector failure, noise, and fiber loss. However, because Eve could use the knowledge of this to refine her eavesdropping, it is safer for Alice and Bob considering all error-bits as due to eavesdropping

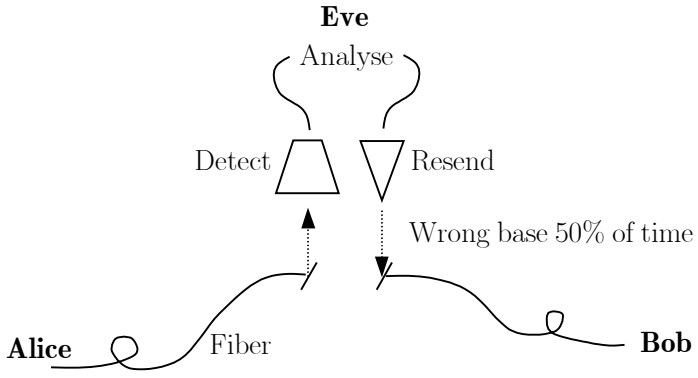


Figure 3.5: Intercept/Resend attack. The fiber is cut and photons are detected and resent by Eve, striving to cover her tracks.

3.4.2 Beamsplit

This attack, beamsplit, is due to the fact that photons are not pure single photon-states. Eve, has a partly-silvered mirror or an equivalent device that diverts a fraction of the light to herself, letting the remaining part pass undisturbed to Bob. See Figure 3.6. In order to avoid wasting information by demodulating pulses in the wrong basis, she stores her pulses until Alice and bob have publicly announced what bases they used. Then she measures the pulses in those bases. There are several types of version of this attack to think of, the most realistic (see [3]) leaks for each successfully demodulated bit at Bob a bit to Eve with probability μ. If the quantum transmission consists of N_s successful pulses, Alice and Bob can estimate that Eve has learned less than

$$v_{beamsplit} = \mu + 5\sqrt{\mu(1-\mu)/N_s}$$

bits of information. The second term is, as before, a standard deviation allowance for Eve having more than average luck in the beamsplitting attempt.

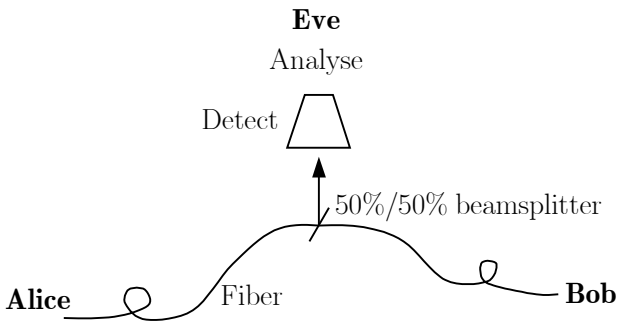


Figure 3.6: Beamsplit attack. Multi-photon qubits are split with a fraction going to Eve. She will measure incoming photons and learn some bits.

3.4.3 Other Attacks

These eavesdropping attacks presented are simple in the way that they act on individual signals only. However, quantum mechanics allow attacks that are more general. Eve can take a helping (auxiliary) quantum system into contact with the signal (signal=bit-information=photon) so that they interact, and then perform a measurement on the helping quantum system to learn something about the original signal. She may also delay the measurement of her helping system in order to take advantage of the information to be exchanged during the public discussion. You can also use several helping quantum systems to probe the whole signal at the same time and from these correlated measurements learn more about the key. This is referred to as *coherent* eavesdropping. A simpler attack probes individual signals (bits) only, and is called *collective* eavesdropping.

Chapter 4

Key Agreement from Shared Information

4.1 The Channels & Authentication

By key agreement we refer to the procedures by which a perfectly secret key is agreed upon between two parties. The key is only to be known by the participating actors, possibly used in further secure communication, e.g. encrypted messages. A key is perfectly secret if any eavesdropper has no other strategy than a random guess for each bit in the key. A block-diagram of a complete protocol for key distribution in quantum cryptography is shown in Figure 4.1. The purpose of this chapter is to explain all steps and concepts shown in this diagram. The raw key (before sifting) is made from bits obtained by Bob after demodulation of incoming photons. Two channels are needed in order to make a key agreement protocol complete.

For transmission of the photons we have the *Private Quantum Channel* (PQC), i.e. an optical fiber. This channel has no authenticity and imperfect privacy. This because an adversary can tamper with and listen to the signals sent, although under certain restriction of quantum mechanics. It should be noted that excessive tampering on the channel can result in suppressing communications between Alice and Bob, but it can not fool them into thinking that they share a secret key when in fact their strings are different or otherwise compromised.

The *Public Channel* (PC) is used by the protocol to exchange *classical* information between Alice and Bob as part of the key agreement. This channel is assumed to have perfect authenticity, but no privacy. This means that Eve can only listen to the channel, but not change the information send on it. Alice and Bob can be sure that the information exchanged between them on the public channel has not undergone any modification, but the entire contents will become known to Eve. The public channel transmits information accurately, possibly because it is supplemented by classical error-correcting code. Newspapers are an example of a secure public channel on which eavesdropping is trivially easy but tampering nearly impossible. The assumption that public messages cannot be corrupted by Eve is an important restriction, because otherwise it is clear that Eve could sit in-between Alice and Bob and impersonate each of them to the other. Eve would then end up with one key shared with Alice and another one with Bob, whereas Alice and Bob would be none wiser. If message authenticity can not be enforced by the physical properties of the channel, it can be provided by a secure classical authentication scheme. In this case a small number of initially secret bits are needed by the protocol, to be used for authentication. Initial secret bits should thus not be necessary for any part of the protocol except perhaps for what is needed to implement this public channel authentication feature.

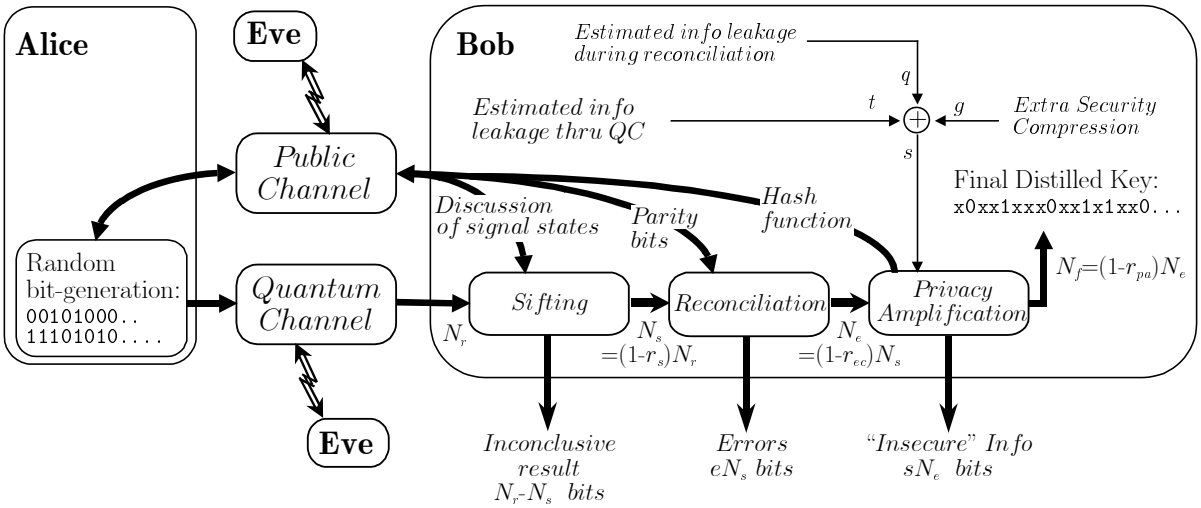


Figure 4.1: Diagram of the basic flow of information in the communication protocol. The raw-key undergoes several stages like sifting, reconciliation, and privacy amplification before the output; a secure key can be shared between both parties. Extensive amounts of bits are thrown away throughout the protocol. Bob estimates t, q, and chooses g, in order to calculate the final compression ratio and level of security.

All steps of the key agreement protocol are shown in the diagram of Figure 4.1. They are in order; sifting, reconciliation (error correction), and privacy amplification. To find a secure key the raw-key has to undergo these three stages, each stage will be explained here in this chapter. We'll refer to this diagram several times.

The public channel is used for the following tasks:

- Making the sifted key; information flow from sender to receiver about which bases were used for each signal.
- Announcement of bit-positions for each block on which the parity is revealed, also exchange of the outcome of all parity-checks.
- For discarding errors; exchange of bits-positions for bits to discard.
- Declaration of the hash-function used in the actual session.
- Optionally: using the initial secret bits to authenticate communication; transmission of an encrypted hashed key.

4.2 Sifting

Alice and Bob use the quantum channel to transmit information. This information is basically a sequence of randomly chosen bits (1:s or 0:s) that each are modulated by a set of quantum states of a photon, which is then transmitted. A set of orthogonal quantum states builds up a base in which either 1 or 0 can be coded. Several schemes to accomplish this, e.g. BB84 and B92, was described is the previous chapter.

The fundamental idea behind the schemes is to publicly exchange information of what bases are used after transmission and reception of the photons. Bob does *not* announce the result of his measurements. Eve will have restricted use of this information because Alice and Bob will agree to use only those bits where Alice and Bob used the same base to modulate and demodulate the photon. However, also for these bits measured in the correct basis, Alice and Bob can not be sure that Eve has not been tampering. Independent of Eve's strategy of eavesdropping she will introduce errors. If she uses the beam-split attack on all photons, she will introduce on average a minimum of 25% errors. As Alice and Bob agrees to use only the bits measured in the correct basis this makes it necessary that Eve will introduce errors if she tries to manipulate the transmission. High error-rates will be detected by Alice and Bob.

The procedure of throwing away bits not measured in the correct basis is called *sifting*. Similarly, they agree to discard those bits where Bobs detector failed to detect any photon, a common event with existing detectors. All these bits thrown away will be classified as inconclusive result.

The resulting *sifted key* is basically the sequence of bits remaining after the inconclusive bits have been removed from the demodulated data, called the *raw key*. The fraction of inconclusive bits r_s depends on the chosen QC-scheme; for B92 and BB84 in average, one half of the received bits are preserved. Thus

$$r_s = \frac{1}{2}.$$

Still, the sifted key contains errors, either introduced by an eavesdropper or arising from noise in the detector. The next step in the process of distilling a perfectly secret key shared between Alice and Bob is to reconcile from the sifted key a new key without errors.

4.3 Reconciliation

Reconciliation is the process of correcting errors between Alice's and Bob's version of the sifted key. Reconciliation is another word for error correction, and applied in order to find identical keys for both parties.

In quantum cryptography, the security relies in the properties of the channel connecting Alice and Bob. Noise is introduced is the physically imperfect channel, and even more errors are introduced by the malicious eavesdropper. The methods implemented and investigated in the simulation program and presented here are based on *public discussion* by exchanging parity bits of the key between two parties. This communication is held over the Public Channel (PC).

There are several, more or less successful, methods to accomplish error-correction efficiently. The method just mentioned makes use of a binary search and parity check procedure. The distinction between error correction and error removing methods should be pointed out. A well-known approach in classical noisy communication introduces redundancy in the signal by use of coding to transmit error-free information. This procedure corrects errors. The usefulness of coding, applied to error correction in quantum cryptography, and it's underlying ideas will be discussed in the first following section. The main problem is that in some way the code words must be known to both sender and receiver, e.g. decided before transmission. A system that has the demand on both parties to share initial secret information prior to key-distribution is rather called a key-growing system. Initial secret information can be seen as the first part of the coming key. For today's implementations coding seem to be only of theoretical interest as experiments aim at making key-distribution systems, but if we have the constraint for Alice and Bob that they must anyway share some initial information then we have a key growing system and coding is natural. However, as will be noticed later, coding can be applied even though no initial information about code-words or such is allowed.

During reconciliation, information is exchanged over the insecure public channel, and thus all information is considered to be known to the eavesdropper. Furthermore, the PC is assumed to be error free by use of standard coded communication. Also, we assume the information transported over the PC cannot be altered by Eve, she can only wiretap it by listening. These properties of the public channel opens up a problem in the reconciliation process; we want to minimize the information that the eavesdropper Eve gains, and at the same time efficiently correct all errors in Bob's key, loosing as small fraction of bits as possible. Why then, must we loose any information?

Unconditionally there is a minimum amount of information that has to be exchanged between Alice and Bob in order to correct Bob's all bits. Either this information can be in form of bits leaked to Eve, or by bits Alice and Bob have to agree to loose in form of introduced redundancy or by discarded bits. This also stand in connection to the Privacy Amplification (PA) in respect to the amount of bits used for security-compression, which will be treated later. We can calculate for this bound in information for a *binary symmetric channel* (BSC) like the Private Quantum Channel (PQC). If Alice and Bob share $n{=}N_s$ bits before reconciliation this corresponds to a knowledge of n bits of Shannon information. Let sender Alice's entropy be denoted $H(A)$, where $A{=}X^n$ is the string of bits sent, and where X is the random process defining either 1 or 0. The entropy of a discrete random variable X is a function of it's p.m.f. (probability mass function), and defined by

$$H(X) = -\sum_{i=1}^{K} p_i \log_2 p_i \ ,$$

where K is the length of the alphabet (in binary case, K=2). See reference [22]. For a binary symmetric channel with probabilities p and 1-p we have the *binary entropy function*

$$H(X) = H_b(p) = -p\log_2 p - (1-p)\log_2(1-p)\,.$$

If each bit send by Alice is randomly and independently chosen it is clear that,

$$H(A) = |A|,$$

where $|A| = n$ and n is the number of bits sent by Alice. If we consider that both the information bits 1 and 0 have equal probability, then

$$H(X) = -\frac{1}{2}\log_2\frac{1}{2} - (1-\frac{1}{2})\log_2(1-\frac{1}{2}) = 1\,,$$

and so

$$H(A) = nH(X) = n\,.$$

The receiver could be either Bob or the less desirable Eve, or both. Let O denote the knowledge picked up any receiver (or observer) of the signal on the Private Quantum Channel. Hence, on the receivers side (or somewhere along the channel) the observed amount of entropy of Alice's send information, given O, will be $H(A|O)$. If we introduce the probability of an error e on the channel, the conditional entropy of X given O is defined by

$$H(X|O) = -e\log_2 e - (1-e)\log_2(1-e)\,.$$

Thus, given the observed knowledge O we can find the mutual information between O and X. $I(X;O)$ will be the amount of information that variable O will provide about variable X, i.e. the received information corresponding to the bits sent by Alice,

$$I(X;O) = H(X) - H(X|O) = 1 + e\log_2 e + (1-e)\log_2(1-e)\,.$$

This is the mutual Shannon information between Alice and the observer, preferable Bob, on the sifted key before error correction and with error rate e. Hence, if O=B, where B is Bob's received bit-sequence, then we will have the shared information between Alice and Bob,

$$I_{AB} = I(A;O = B)\,.$$

If O represents the knowledge from Eve's measurements in action of eavesdropping, having bit-string E, we have

$$I_E = I(A;O = E)\,.$$

Now, we are ready to calculate the minimum amount of information to be exchanged.

After that Bob have received Alice's string of bits he will have N_s bits. Only a fraction I_{AB} of these will be correct; $N_s I_{AB}$. The difference N_s - $N_s I_{AB}$ will thus be the

amount of information corresponding to the information content in the error bits occurring with a rate of e. Thus

$$N_s(1-I_{AB}) = H(A|O=B) = N_s H(X|O).$$

The minimum number of exchanged (lost) bits is therefore

$$N_{\min} = N_s(1-I_{AB}) = N_s(-e\log_2 e-(1-e)\log_2(1-e)).$$

In other words, Eve's information is the *change in entropy* of Alice's string A upon it's way transferred, received, and reconciled by Bob, giving $B'=A$.

The question is now how close we can be to this limit in an error correction protocol. Obviously, we want to loose as little information as possible, therefore N_{min} sets a lower limit. We will see that the last protocol **cascade**, will work very close to this Shannon limit for large enough N_s. It is not computationally efficient compared to coding, but nevertheless saves more of the fragile bits of the key. It is most important to keep the bit-rate as high as possible through every step in the QC-system.

The performance of error-correction in terms of how close it works to the Shannon limit will be investigated. It is not obvious that all errors will be corrected, this depends on the correction algorithm. Usually several iterations of the algorithm are necessary. There is a risk with too many errors that not all errors can be corrected at all within a limited number of iterations. An important parameter is the maximum error-rate e_{lim} (QBER), where below, all errors are corrected. Different approaches of reconciliation can be more or less efficient, (time consuming). We can investigate how many iterations are needed to leave the reconciliation without errors remaining. In other words, what is the success-rate?

It should be remembered that, of course there can be two types of undesired errors, namely noise in channel and errors introduced by an eavesdropper. These errors cannot be distinguished from each other, and so we must assume that they arise from Eve and that the information she has about the key before error correction is at least all the error-bits. During reconciliation she will gather additional information by wiretapping the public discussion, and we must keep this leakage low. After reconciliation define the key-string shared between Alice and Bob by S. Let Q denote the amount of information exchanged on the PC. Then $I_E(S|Q)$ is the expected amount of Shannon information that an eavesdropper can get on S given Q. We will assume that Alice and Bob now share identical keys with a certain probability. We will also assume that has knowledge of all error-positions.

In the case of error correction the key-length will remain the same, $N_r=N_s$, after reconciliation. In the case of error removing; $N_r<N_s$. Common for both ways of reconciliation is that Alice and Bob will learn the actual error-rate e (i.e. the $QBER$) used later for privacy amplification.

To summarize: we investigate how the use of a channel with perfect authenticity (PC), but no privacy can be used to restore the errors from a transmission over a channel with no authenticity and imperfect privacy (PQC).

We will now explain how coding can be used, and why it can be questioned in the context of quantum cryptography error-correction.

4.3.1 Classical Coding - Useful?

In classical communication, linear block codes, cyclic codes, and convolutional codes, are applied as a standard approach of error correction [13],[1]. In quantum cryptography error-correcting codes is not quite adequate because it is based on

assumptions that a few errors are more likely than many, and that errors are not spitefully introduced by an rival. With today's performance of detectors and fibers for experimental quantum cryptography we have a quite high error probability - too high for convolutional coding to make justice. Remembering that Eve has the possibility to tamper with the bits in the Private Quantum Channel, we indeed realize that codes are susceptible to a very simple threat. Eve can just replace the bits sent by Alice with bits by her choice, and thus she can suppress the communication entirely injecting any other code word of her choice, instead of the real one. This can however be impeded by sending the coded information on the public channel after the real bit-sequence is already send over the PQC. In this *post-facto* application way, Eve does not have the advantage of knowing the code words at the time of the bit-sequence transmission.

So, if Eve in a-priori knows the code-words to be used, classical coding error correction would not be possible. But of course, this would work in the case when Alice and Bob initially share secret information, for instant the code-words, then we refer to the key-distribution as a key-growing scheme. If we cannot let Alice and Bob initially share any secret information, or if we assume that Eve have knowledge of the secret code words, then this traditional *non-post facto* coding manner would not be applicable. All these arguments has to be treated in the discussion of authentication.

A solution is for Alice to randomly choose a code, send the coded information, representing the information in x, on the PQC and wait for Bob to publicly announce that he received the information. First then she announces the used code word on the public channel (where Eve cannot alter it) and Bob can decode and error-correct his information. In this way Alice reveal information about the code-words only when it is too late for Eve to have any practical use of it, altering the private channel transmission. Another criteria that motivates this post-facto application is the sifting procedure. Here we randomly use only half of the bits, on average, because of the wrong-base bit measuring. We do not in advance now which bits will fall away in this step. Therefore coding has to be applied after sifting.

Although coding is a workable way, it remains true that classical coding assumes low error-rates, and can therefore not be applied efficiently to correct errors in a quantum cryptography transmission. We will now calculate e_{lim} applying a typical convolutional code. We calculate the bit-rate R_{ec} vs. the $QBER$, e. For this we must find the bit-loss of coding. To find the compression level of privacy amplification we need to estimate the information q lost to Eve during reconciliation. The information-loss for error-correction is related to the coding capacity for convolutional coding C_{cap}.

Let us briefly go through an example of coding-applied error-correction (Bennett et al [6]). After transmission of Alice bit-sequence x to Bob over the private channel, Alice and Bob first agree to uniform their keys x, respective y, i.e. apply a random permutation on the key to randomly distribute all errors. Alice applies the codes W to x and transmits the result $W(x)$ to Bob over the public channel, thereby giving away at most $|W(x)|$ bits of information to Eve about x. Bob uses $W(x)$ to correct all errors in y to recover x. This can be done if the code W is sufficiently effective to correct all errors. If $W(x)$ is considerably shorter than x, there is still information that Eve does not know about x and the method of privacy amplification can be applied to distill a secret key by compressing x with an amount of $|x|$-$|W(x)|$, where Eve will have at most one bit of information on x shared between Alice and Bob.

In a traditional non-post facto situation where both x and W are send over the public channel the capacity C_{cap} of the transmission is given by

$$C_{cap} = \frac{|x|}{|x| + |W(x)|} = \frac{k}{k+n},$$

where k is the length of x, k=$|x|$ and n is the length of the coded information $|W(x)|$, n=$|W(x)|$. The code-rate R_c is defined as

$$R_c = \frac{k}{n}.$$

Shannon has showed the theoretical capacity remains the same for the post-facto coding case. In the same way it can be shown that the effective capacity C_{cap} achievable by convolutional coding is still the same. It is given by the expression

$$C_{cap} = H(X) - \log_2(1 + 2\sqrt{e(1-e)}) = \frac{k}{k+n} = \frac{R_c}{1+R_c},$$

and we can solve for the maximum information leaked to Eve during transmission, n=$|W(x)|$. For a BSC we know that $H(X)$=1, hence

$$n = |W(x)| = k\left(\frac{1}{1 - \log_2(1 + 2\sqrt{e(1-e)})} - 1\right).$$

Let q be the fraction of bits lost to Eve, then we have

$$q = \frac{n}{k} = \frac{1}{1 - \log_2(1 + 2\sqrt{e(1-e)})} - 1.$$

Knowing C_{cap} we can find the fraction of bit-rate decrease due to coding-error correction r_{ec}, and knowing q we can form the fraction r_{pa} of bits compressed by privacy amplification due to error-correction information leakage.

$$r_{ec} = 1 - C_{cap} = 1 - (H(X) - \log_2(1 + 2\sqrt{e(1-e)})) = \log_2(1 + 2\sqrt{e(1-e)}),$$

and

$$r_{pa} = q = \frac{1}{1 - \log_2(1 + 2\sqrt{e(1-e)})} - 1.$$

Now the expression for the maximum normalized bit-rate R_{ec}, can be set up. This is actually the achievable rate using convolutional coding for error correction, and with privacy amplification based on the information loss of this method.

$$R_{ec}(e) = (1 - r_{ec})(1 - r_{pa}) = (1 - (1 - C_{cap}))(1 - q) =$$
$$(1 - \log_2(1 + 2\sqrt{e(1-e)})) \cdot \left(2 - \frac{1}{1 - \log_2(1 + 2\sqrt{e(1-e)})}\right)$$

$R_{ec}(e)$ is plotted in Figure 4.2. We find the value of e_{lim} to be about 4.5%. Coding can not be used for error-rates exceeding this value. In other words, it will be guaranteed

to correct all errors up to this error-rate limit. The capacity C of the private quantum channel (PQC) is shown for comparison, and is given by

$$C = (1 - r_s)(1 - H_b(e)) = \frac{1}{2}(1 + e\log_2 e + (1-e)\log_2(1-e)) .$$

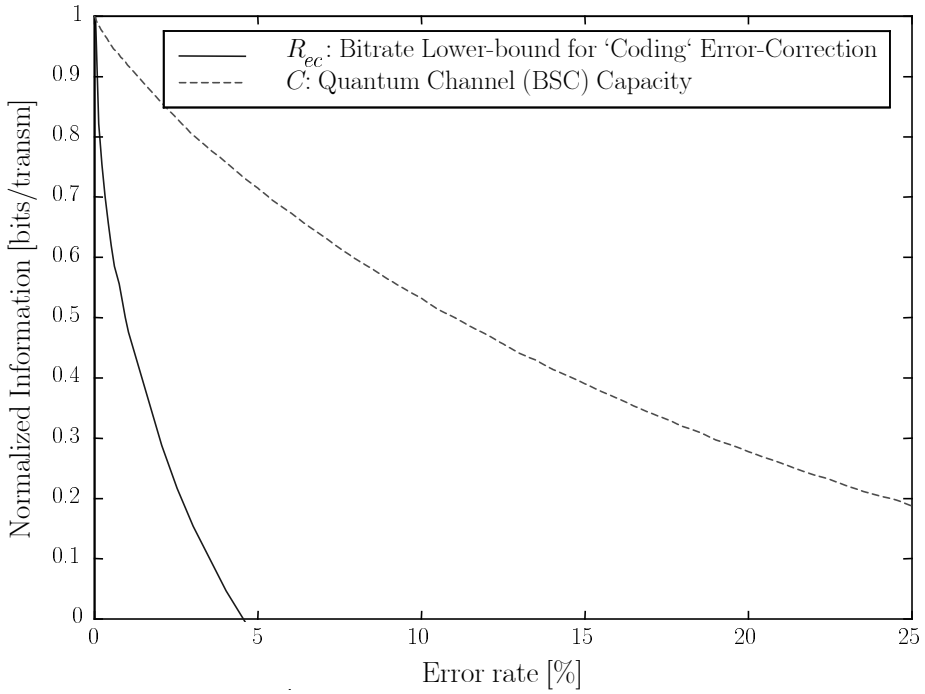


Figure 4.2: The solid line represents an lower bound on the performance for *convolutional coding* error-correction in terms of normalized Shannon information (bit-rate). The method will guaranteed correct all errors up to an error-rate limit e_{lim} of 4.5%. The dotted line is the capacity for a binary symmetric channel (BSC) like the PQC, shown here for comparison.

It would be of interest to find codes approaching the limit of the Shannon capacity and in this way be comparable to the cascade method.

We now turn our interest to a more simple procedure of binary search and parity checks, as a complement to coding.

4.3.2 Binary Search and Parity Check - Discard Errors

In the following three sections, three procedures of error *correction* are presented. They are all variations of the simple error correction method where you compare the parity (XOR) of a subset of both versions of the bit-sequence. If the parities are not matching then you know an error occurred, and you proceed with a binary search on that subset with subsequent parity checks to find the error. Each subset will be left with an even number of errors or none. This procedure can be applied several times proceeded by a random permutation of the bits, to randomly shuffle around the error locations.

This way of error correction can be related to linear block-codes and parity check matrices. The difference in our case would be that the parities are exchanged only

after raw-data transmission, and not together with the raw-data. Compare this with the post- and non-post-facto situation for coding discussed in the previous section. Applying parity check matrices and sending them with the information would be inefficient in the sense that Eve then could acquire unnecessarily much of information. Sending parity check matrices through the private quantum channel would be like coding and will not give such a high bit-rate. Sending the parity check matrices over the public channel would give Eve knowledge of the whole check matrix. A better way is therefore for Alice and Bob to compare blocks of their bit-sequences, and based on this result Bob announces to Alice which parity-bits he needs for further error correction. In this way, we will keep down the amount of information sent over the public channel.

The method of this section rather discard errors instead of correcting. This is mainly the bisect-and-discard protocol that has been proposed [3]. The argument for Alice and Bob to agree about discarding bits is that Eve would then not gain any knowledge from the public channel discussion. Remembering that Eve will collect all information send over the public channel, each time a parity bit is checked, Eve will have also have that bit. In order to reduce Eve's benefit of knowing this parity bit, Alice and Bob agree to discard one bit in the corresponding block. This reduces her information of the other bits in the block to zero. The disadvantage of discarding errors is that Alice's and Bob's key will be shortened during reconciliation. Yet, this can be compensated by the fact that no information is revealed to Eve and thus no extra privacy amplification compression for this reason is needed. The idea of the next method arise from here, where information is lost to Eve through public discussion, but where the key-length remains the same after reconciliation. Can the final bit-rate be made higher allowing more compression with no key shortening than no compression but with key shortening? This is also discussed in the section of privacy amplification.

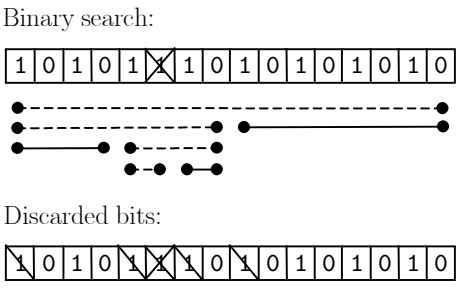


Figure 4.3: The *discard* error-correction by binary search and parity disclosure on a chosen sub-block of the key. The error-bit (x-marked) is identified by further bisective block-divisions, with public parity comparison as shown. The solid line represents a matched parity-check, and the dashed line a miss-match. This procedure of error correction reminds of the classic 'divide and conquer' strategy. Also shown are the remainder bits after some have been deleted.

Now will follow a description of the protocol "binary search and parity check, removing errors." Assume that Alice and Bob before reconciliation each have N_s bits, where a fraction e of these are in error for Bob. Then we have a total of eN_s errors.

1. Perform a random permutation of the sifted bits. That is, randomly shuffle the errors to deal with eventual error-bursts.

2. Divide the N_s bits into blocks of size $p=N_s/eN_s=1/e$ bits. Then we will have in average one error in each block. Compute the parity of the block on both Alice's and Bob's side. Compare all parities. If the parity match, discard one bit from that block, and go on to the next block. If the parity miss-match then a further bisection is undertaken such that the block is divided into two sub-blocks, the parity of one sub-block is checked against the other party. Only one sub-block will now have parity match, so discard one bit from that sub-block and do further bisection into two halves with parity check on the sub-block with miss-matching parity. Continue in this way until only two bits are left in one sub-block, then both those bits will be discarded. See Figure 4.3.

3. Now all blocks are examined and we now that each block contains even number of errors or none. To remove the remaining errors, repeat the procedure above; steps 1 and 2. The number of iterations for this is based on the error-rate e estimated in the beginning. For each iteration, increase the block-size to at most $2N_s$.

All other protocols explained later in this chapter are based on the following primitive interactive protocol that finds and discards an error in a given string of bits. When string A and B has odd number of errors, interactive binary search can find an error by exchanging fewer than $\log_2 n$ bits over the PC.

Algorithm 4.1:

Binary:

1: Alice has bit-sequence $A \in \{0,1\}^N$, Bob has bit-sequence $B \in \{0,1\}^N$. *Bob* receives from *Alice* the parity of the first half of the string A.

2: *Bob* determines whether an odd number of errors occurred in the first half of or in the second by comparing the parity (XOR sum) of the first half of his string B with the parity send by *Alice*.

3: This process is repeatedly applied to the half determined in step **2:** An error will be found and discarded! On all halves with even number of errors one bit is agreed to be discarded.

The discard-protocol is explained further with more details in the pseudo-code of Algorithm 4.2. When examining this error-correction method it becomes obvious that a lot of valuable bits are lost by discarding bits from the key, while no information is lost to the eavesdropper. As a consequence we have Eve's information gain q due to error-correction equal to zero, $q=0$. Furthermore, how large is the fraction r_{ec} by which the key is shortened? Is there any upper bound? Tancevski et al, derives in [31] a formula for this limit,

$$r_{ec} = \frac{7}{2}e - e\log_2 e\,.$$

We want to derive the normalized bit-rate R_{ec} for this error-correction method, just like in the previous section of coding. We have for privacy amplification

$$r_{pa} = q = 0\,,$$

and we can now write

$$R_{ec}(e) = (1 - r_{ec})(1 - r_{pa}) = 1 - (\frac{7}{2}e - e\log_2 e).$$

Algorithm 4.2:

Discard protocol:

1: Alice has bit-sequence $A \in \{0,1\}^N$, Bob bit-sequence $B \in \{0,1\}^N$.
2: Bob choose block-length $p_{j-1}=1/e$ where e is the $QBER$. j=1.
3: **While** $V_e \neq \emptyset$ **Do:**
- **3.1:** Pass j: Bob determines a new function $f(X)$ that has the property of randomly permuting the bits of an input bit-sequence X.
- **3.2:** Bob send f to Alice. Alice computes $f(A)$ and Bob, $f(B)$.
- **3.3:** Alice and Bob independently determine the k_j= floor(N/p_j)+1 blocks by subsequently choosing $p_j=2p_{j-1}$ ($p_1=1/e$) bits from A and B. Let $\mathcal{B}$ be the set of bits in K.
  For $i = 1...k_1$:
  - Alice and Bob make block $V_i \in \mathcal{B}$ by choosing p_j random bits starting with the first bit in each sequence from the key K. Let $\mathcal{B}_i$ be the set of bits in V_i, $\mathcal{B}_i \subseteq \mathcal{B}$.
  - Subtract these bits from the set of bits $\mathcal{B}$ in key K, $\mathcal{B}' = \mathcal{B} \setminus \mathcal{V}_i$ to make $K \in \mathcal{B}'$.
  - Make block $V_{i+1} \in \mathcal{B}'$ by again choosing the first p_i bits from K.
  - **If** $i = k_1$, $\mathcal{B}'$:$\{0,1\}^p$ where $p<p_j$ **then** $V_{i+1}=R$, $R \in \mathcal{B}'_p$.
- **3.4:** Bob send parity of all blocks V_i, $i = 1...k$ to *Alice*. Bob receives information which blocks V_e were in error (i.e. blocks with odd # of errors).
  Bob runs **Binary** on each error-block V_e to remove the error. In all other blocks Bob deletes the first bit in pos. l for that block. Bob sends all l:s to Alice and she does the same.
  - $j=j+1$.

4: **End:** All errors found!

$R_{ec}(e)$ is plotted in Figure 4.4. We find a value on e_{lim} of about 16.5%. No bits would remain in the final key for error-rates exceeding this value. The capacity C is shown for comparison.
We will take a look at what happens if we *correct* the errors we find, instead of delete them. The change is that Eve now will gain knowledge by using the information she learned by listening to the public channel communication. This is, q will be bounded away from zero.

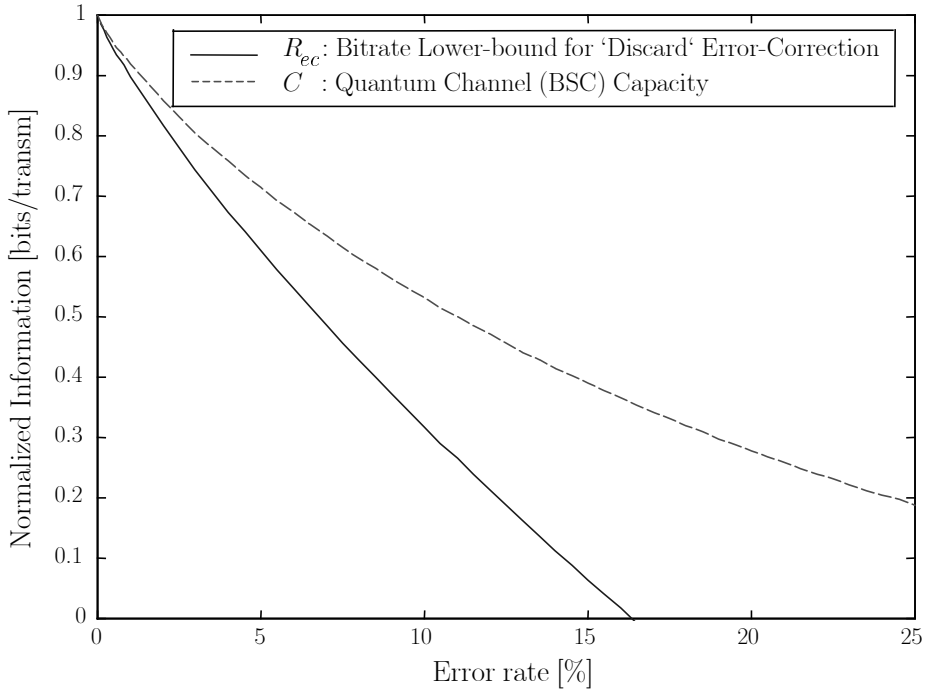


Figure 4.4: The solid line represents an lower bound on the performance for the *discarding errors* error-correction, in terms of normalized Shannon information (bit-rate). The method will guaranteed correct all errors up to an error-rate limit e_{lim} of 16.5%. The dotted line is the capacity for a binary symmetric channel (BSC) like the PQC, shown here for comparison.

4.3.3 Binary Search and Parity Check - Correct Errors

This method is very much the same as the previous. The difference is that here we correct the errors found instead of deleting them. The procedure is a modification of the bisect-and-discard protocol that was proposed in [3], based on disclosing the parity of blocks of bits and discarding bits so that the errors are corrected. If we now do not discard bits, we have to calculate for Eve's learnt information about the key from the public channel parity-bit exchange. Later we have to remove this fraction of information from Alice's and Bob's key in the privacy amplification compression part.

The question is now; can this compression in some way be compensated by the fact that the key in this case will not be shortened, but remain with the same number of bits after reconciliation as before. It turns out that the key-rate will almost be the same. Simulations have shown that there only a slight difference between the two approaches. In fact discarding errors is better.

The pseudo-code in Algorithm 4.3 explains the procedure of correcting errors. For a brief overview, we refer to the last section and the description of discarding errors. The procedure is almost the same, except that discarded bits (also in **Binary**) should be left unchanged and the found error-bits should be corrected (i.e. flipped). See Figure 4.5.

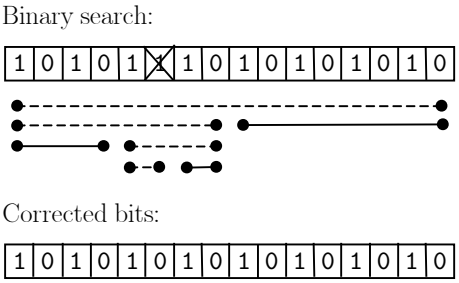


Figure 4.5: The *correct* error-correction by binary search and parity disclosure on a chosen sub-block of the key. The error-bit (x-marked) is identified by further bisective block-divisions, with public parity comparison as shown. The solid line represents a matched parity-check, and the dashed line a miss-match. Also shown are the remaining bits after correction.

Algorithm 4.3:

Correct protocol:

1: Alice has bit-sequence $A \in \{0,1\}^N$, Bob bit-sequence $B \in \{0,1\}^N$.
2: Bob choose block-length $p_{j=1}=1/e$ where e is the $QBER$. $j=1$.
3: **While** $V_e \neq \emptyset$ **Do:**
- **3.1:** Pass j: Bob determines a new function $f(X)$ that has the property of randomly permuting the bits of an input bit-sequence X.
- **3.2:** Bob send f to Alice. Alice computes $f(A)$ and Bob, $f(B)$.
- **3.3:** Alice and Bob independently determine the k_j= floor$(N/p_j)+1$ blocks by subsequently choosing $p_j=2p_{j-1}$ $(p_1=1/e)$ bits from A and B. Let $\mathcal{B}$ be the set of bits in K.
  For $i = 1...k_1$:
  - Alice and Bob make block $V_i \in \mathcal{B}$ by choosing p_j random bits starting with the first bit in each sequence from the key K . Let $\mathcal{B}_i$ be the set of bits in V_i, $\mathcal{B}_i \subseteq \mathcal{B}$.
  - Subtract these bits from the set of bits $\mathcal{B}$ in key K , $\mathcal{B}' = \mathcal{B} \setminus \mathcal{V}_i$ to make $K \in \mathcal{B}'$.
  - Make block $V_{i+1} \in \mathcal{B}'$ by again choosing the first p_j bits from K .
  - **If** $i = k_1$, $\mathcal{B}'$:$\{0,1\}^p$ where $p<p_j$ **then** $V_{i+1}=R$, $R \in \mathcal{B}'_p$.
- **3.4:** Bob send parity of all blocks V_i, $i = 1...k$ to *Alice*. Bob receives information which blocks V_e were in error (i.e. blocks with odd # of errors).
  Bob runs **Binary** on each error-block V_e to correct the error.
  - $j=j+1$.

4: **End:** All errors found!

The problem of the current method is how to calculate for the raw information leaked to Eve. Let us for simplicity assume that the fraction q, estimating this information will be given by the number of bits discarded in the previous method

$$q = \frac{7}{2}e - e\log_2 e\,.$$

No bits will be thrown away, thus

$$r_{ec} = 0\,.$$

We'd now like to derive the normalized bit-rate R_{ec}. We have for privacy amplification the compression

$$r_{pa} = q = \frac{7}{2}e - e\log_2 e\,.$$

From these two fractions we can form the performing bit-rate

$$R_{ec}(e) = (1 - r_{ec})(1 - r_{pa}) = 1 - (\frac{7}{2}e - e\log_2 e).$$

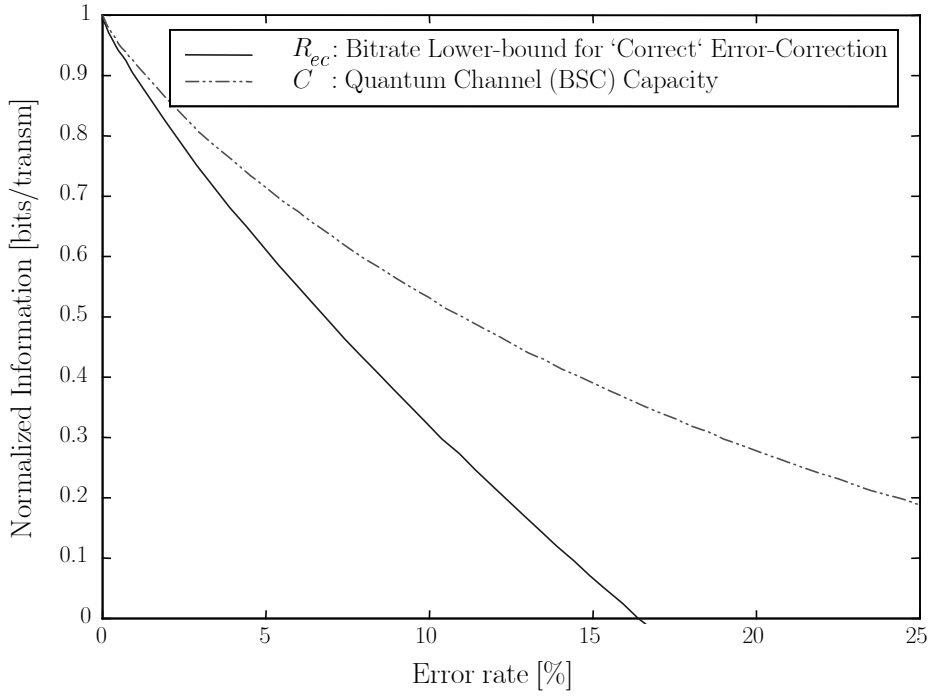


Figure 4.6: The solid line represents an lower bound on the performance for the *correcting errors* error-correction, in terms of normalized Shannon information (bit-rate). The method will guaranteed correct all errors up to an error-rate limit e_{lim} of 4.5%. The dotted line is the capacity for a binary symmetric channel (BSC) like the PQC, shown here for comparison.

$R_{ec}(e)$ is plotted in Figure 4.6. We find a value on e_{lim} of about 16.5%. All bits in the reconciled key is compressed for error-rates exceeding this value. The capacity C is shown for comparison.

Finally, going back to the comparison with the previous method; discarding errors, we can argument for the advantages versus the disadvantages between the two methods. For discarding errors: for every iteration with block-division the key-length decreases and gives less binary search and parity-checks. Thus there are less unnecessarily removed bits for every iteration. Here is a benefit over the correcting errors method. The discarding-errors method will therefore as a total have removed less errors than Eve has learned from the public discussion in the correcting-errors method. Discarding errors is for that reason only slightly better. Of course this depends on how q is estimated.

4.3.4 Cascade - ad hoc Binary Search

A better reconciliation algorithm has been proposed. It is basically a modified version of correcting errors by binary search and parity checks. It method was first presented by Brassard, Salvail in [9] as known as the **cascade**. They prove that the capacity for reconciliation-processes, belonging to this class of optimal-schemes, approaches the Shannon limit. The cascade method is designed for practical implementations and is not optimal. Yet, the capacity is very close to that of noisy channel coding. The basic change of idea compared to previous protocols is to now remember the error location for each error found in an iteration, and use this when going back to all previous passes to correct even one more error. For every error found a new error will be found and corrected. Remembering that **binary** will leave each block with even numbers of errors or none we will see how this works. First we define a *pass* as the starting point of the a new iteration that will randomly divide the key in a new set of blocks for further error-search. Cascade works in several passes.

So, in every pass each time an error is found in a block, it's corresponding error-position will also be found in a block of a previous pass.

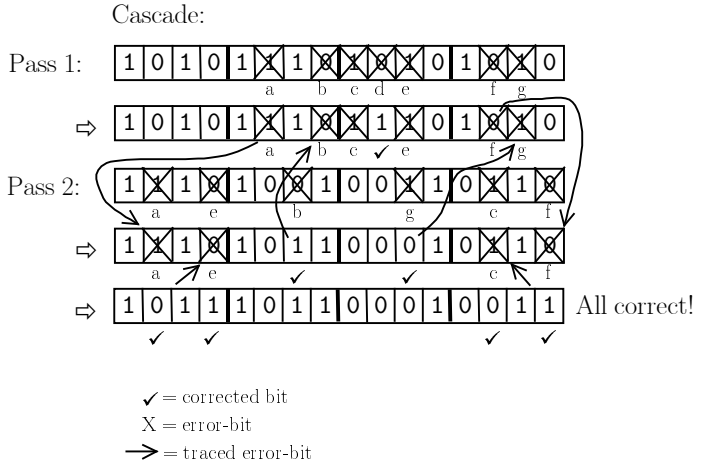


Figure 4.7: The *cascade* error-correction by binary search and parity disclosure on a chosen sub-block of the key. Error d is the only one found and corrected by **Binary** in Pass 1. A random permutation of the key is then carried out, prior to entering Pass 2. In this stage two errors, b and g are found, but now looking for these corrected errors and their positions in the stage of Pass 1, we find even two

more errors, a and f to correct. We can run **Binary** on the first and the last block in the final stage to correct the last two errors, e and c.

Algorithm 4.4:

Cascade protocol:

1: Choose block-length p_1, based on the $QBER$.

2: Pass j=1: Make a temporary key K_T^j with k_j=floor(N/p_j)+1 blocks of smaller subsets V_i, $i = 1...k_1$ from the original key $K \in \{0,1\}^N$ in the following way. Let $\mathcal{B}$ be the set of bits in K.
For $i = 1...k_1$:
- Make block $V_i \in \mathcal{B}$ by choosing p_j random bits from the key K. Let $\mathcal{B}_i$ be the set of bits in V_i, $\mathcal{B}_i \subseteq \mathcal{B}$.
- Subtract these bits from the temporary set of bits $\mathcal{B}$ in key K, $\mathcal{B}' = \mathcal{B} \setminus \mathcal{V}_i$.
- Make block $V_{i+1} \in \mathcal{B}'$ by choosing p_j random bits from the now smaller temporary key.
- **If** $i = k_1$, $\mathcal{B}' : \{0,1\}^p$ where $p < p_j$ **then** $V_{i+1} = B$, $B \in \mathcal{B}'_p$.

3: Send parity of all blocks V_i, $i = 1...k$ to *Alice*. Also send the bit-positions $l \in K_T^1$ for each block V_i. Then *Alice* will also know the blocks.

4: Alice compare Bobs parities with hers. She returns information about which blocks were in error V_e (i.e. blocks with odd # of errors).

5: *Bob* runs **Binary** on each error-block V_e.

6: Pass j>1:
Do step **2:** to **5:** with $j = j+1$ and $p_{j+1} = 2p_j$.
In step **4**, errors in bit-position $l \in K_T^j$ will be found for some blocks $V_e : \{V_i^j, 1 \le i \le k_j\}$.
If $V_e \neq \emptyset$,
- Correct all error-bits l in V_e. All the blocks V_i^u for $1 \le u < j$ (blocks in previous passes) such that $l \in K_T^j$ will now have a odd number of errors. Let $\mathcal{V}$ be the set of these blocks.
- *Bob* runs **Binary** on one block in $\mathcal{V}$ corresponding to the smallest u, and corrects one additional error. Let $l' \in K_T^j$ be the position of this error origin from block(s) V_i^u from each pass u=1..j. Let $\mathcal{E}$ be the set of these blocks.
- Compute $\mathcal{V}' = \mathcal{E} \Delta \mathcal{V} = (\mathcal{E} \cup \mathcal{V}) \setminus (\mathcal{E} \cap \mathcal{V})$, i.e. the set of blocks for u=1..j containing odd number of errors and repeat previous point followed by computing $\mathcal{V}' = \mathcal{E} \Delta \mathcal{V}$ until $\mathcal{V}' = \emptyset$. Then, all blocks V_i^u, u=1..j will after completion of pass j have an even number of errors or none.
- **Repeat** step **6**

Else break!

7: **End:** All errors found!

This block was previously left (after binary search) with an even number of errors, but now as the actual error is corrected we note that this block has an odd number of errors (it cannot have none!). Binary search is applied on that block again, and one more error will be found. This new error may in it's turn also have a corresponding error in an other block that can be corrected in the same way. This process is continued until no more errors are found for that pass. The number of passes needed is determined by Alice and Bob before execution, based on the error-rate e. It is shown optimal that the block-sizes are doubled for each iteration, $p_{j+1}=2p_j$.

How all this works, can easiest be understood by turning directly the protocol description, see Algorithm 4.4:. See also Figure 4.7.

We now like to derive the bit-rate for this protocol, to see how close it is the the Shannon limit. The formula for q can be derived from [9] as a bound on the maximum information leaked. It is given by

$$q \leq 2 + \frac{1-(1-2e)^{0.73/e}}{2}\log_2\left(\frac{0.73}{e}\right) + \left(0.73 - \frac{1-(1-2e)^{0.73/e}}{2}\right)\cdot\log_2\left(\frac{0.73}{e}\right)\cdot\sum_{l=2}^{w}2^{1-l} \ .$$

where w is the number of passes to reach zero error-rate after reconciliation. $p_1=0.73/e$ is the block-size in bits used in the first pass $j=1$. The value of 0.73 has been experimentally found good for the first pass.

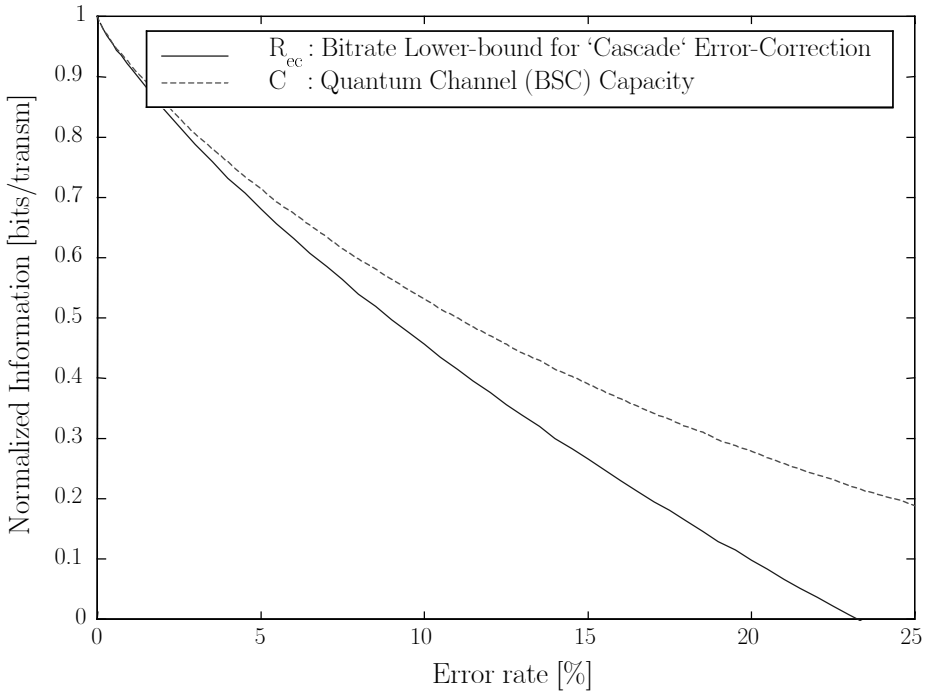


Figure 4.8: The solid line represents an lower bound on the performance for the *cascade* error-correction, in terms of normalized Shannon information (bit-rate). The method will guaranteed correct all errors up to an error-rate limit e_{lim} of 23%. The dotted line is the capacity for a binary symmetric channel (BSC) like the PQC, shown here for comparison.

We now end by calculating the normalized bit-rate R_{ec}. We have a constant number of bits through the reconciliation process because no bits will be thrown away. Thus

$$r_{ec} = 0 .$$

We have for privacy amplification the compression

$$r_{pa} = q .$$

Using these two fractions we can write,

$$R_{ec}(e) = (1 - r_{ec})(1 - r_{pa}) =$$

$$-1 + \frac{1-(1-2e)^{0.73/e}}{2} \log_2\left(\frac{0.73}{e}\right) + \left(0.73 - \frac{1-(1-2e)^{0.73/e}}{2}\right) \cdot \log_2\left(\frac{0.73}{e}\right) \cdot \sum_{l=2}^{w} 2^{1-l}.$$

$R_{ec}(e)$ is plotted in Figure 4.8. We find a value on e_{lim} of about 23%. All bits of the final key shared by Alice and Bob would be compressed for error-rates exceeding this value. The capacity C is shown for comparison.

This value is extensionally higher than for other protocols presented, and simulations will show that this protocol is fact works closer to the capacity as expected. Less information is leaked. The question is how to choose the starting block-length p_1 for optimal reconciliation-performance. Usually good performance is achieved by choosing p_1 so that each block contain one error.

We now turn our interest to the next step; privacy amplification where you compress the key by the amount of bits (information) that Eve is expected to have acquired during error-correction, plus the amount of information gained by eavesdropping on the quantum channel.

4.4 Privacy Amplification

The fundamental problem in quantum cryptography is to distribute a perfectly secret key between two parties, Alice and Bob, even though an eavesdropper Eve can collect partial knowledge about it. Privacy amplification is the art of distilling highly secret information shared between two parts from a larger set of information that is not fully secret. For this is used so called *universal hash-functions*, first introduced by Carter and Wegman (1979). Privacy amplification by public discussion was first introduced by Bennett, Brassard and Robert in [6] and [7] (1988). Many other papers cover this topic, e.g. [5],[11]

We wish to reduce Eve's information on the key to an arbitrarily low amount by compressing the key. Let Alice and Bob share a random variable V_e , e.g. a n-bit long string, while the eavesdropper learns a correlated random variable Z about V_e. Z provides at most $t<n$ bits of information, i.e. $H(V_e|Z) \geq n\text{-}t$. The exact eavesdropper distribution $p_{V_e|Z}$ is generally unknown to Alice and Bob. Alice and Bob now wish to agree on a publicly chosen random compression function f: $\{0,1\}^n$ - $\{0.1\}^r$ and calculate $f(A)$ respective $f(B)$, (note: $A=B$). Eve's partial information Z on V_e and her complete knowledge of f will give her arbitrarily information about $f(Z)$. The size of r will depend on amount of information available to Eve as well as, but to a surprisingly small extent, on the kind of information. Variable g will be the security compression parameter restraining the amount of information known by Eve on the final key.

The elegant result is that for any $g > 0$ Alice and Bob can distill a fraction of $r=n\text{-}t\text{-}g$ bits long secret key while keeping Eve's expected information about $f(A=B)$ smaller than $2\text{-}gN_e/\ln 2$ bits. This is quite remarkable since Alice and Bob don't know

$p_{V_e|Z}$. The function f is chosen from a set of universal hash functions, defined down below. It is the complete chaotically property of these function that will accomplish the result. Note again that f will be known to Eve.

Let $K= f(A=B)$ be the shared r-bit long key after privacy amplification. In general, f is selected randomly from an appropriate set of functions F, in order to avoid that Eve knows f before deciding about her information knowledge about V_e. Interesting are the bound on $I(K;F,Z)$. By corollary 5 in [7] the following is stated:

Corollary: *Let V_e be a random n-bit string with uniform distribution over $\{0,1\}^n$, let $Z=x(V_e)$ for an arbitrarily eavesdropping function x: $\{0.1\}^n \rightarrow \{0.1\}^t$ for some $t<n$, let $g<n-t$ be a positive safety parameter, and let $r=n-t-s$. If Alice and Bob choose $K=F(V_e)$ as their secret key, where F is chosen at random from a universal class of hash functions from $\{0.1\}^n$ to $\{0.1\}^r$, then Eve's expected information about the secret key K, given F and Z satisfies*

$$I(K;F,Z) \leq \frac{2^{-g}}{\ln 2}.$$

In other words, this formula will be an upper bound for Eve's information on Alice's and Bob's shared bits after privacy amplification. It can be made arbitrarily small by choosing g large enough. Note that this requires no information about the $QBER$ e, you need only to know an upper bound for t.

What is then a universal hash-function?

> **Definition:** A class $\mathcal{F}$ of functions $\mathcal{A} \rightarrow \mathcal{B}$ is universal if, for any distinct y_1 and y_2 in $\mathcal{A}$, the probability that $f(y_1) = f(y_2)$ is at most $1/|\mathcal{B}|$ when f is chosen at random from $\mathcal{F}$ according to the uniform distribution.

For short, you can say that a hash-function randomly redistributes the bits, the output bits are a chaotic permutation of input bits, that is, a small input-change will provide a large output-change. When picking a function f uniformly at random from this class, elements (bits) are mapped independently, and the image of each element is uniformly distributed. It's a sort of one-way function, e.g. if Eve know the whole key except one bit, then the output will look completely random compared to the output where the input-bits are the correct key. In other notation we have for the hash-function $h(x)$: $\{0.1\}^{N_r} \rightarrow \{0.1\}^{N_f}$:

$$N_f = N_r - tN_s - qN_s - gN_e.$$

Thus, the key is compressed by a fraction of $r_{pa}=t+q+g=u+v+q+g$ bits. For a discussion on how to estimate of the information leaked to Eve, t, see section 3.4 and 4.3.

The hash-function used in the implementation of the simulation program of chapter 6 is a simple XOR-multiplication of the key with a random binary matrix. This is an example of the class H_3 universal linear hash-functions [9] that requires $N_e \times N_f$ size matrices containing random 1:s and 0:s. There are several classes of hash-functions to use. Universal hash-functions in general, requires $N_e \times 2^{N_f}$ size matrices. Of course computations are made faster with the use of smaller matrices, and there are other classes of hash-functions to achieve these desires, but the class of linear functions used here are however easy to implement.

As a summary can be said that privacy amplification will, as the last stage in quantum cryptography protocol, leave an arbitrarily secure key shared by Alice and

Bob just as desired. The level of security depends only on the choice of g. We will now investigate the effective bit-rates achievable by different reconciliation techniques, and also with consideration taken to the decrease of bit-rate in the steps of sifting and privacy amplification. Review Figure 4.1.

4.5 Quantum Bit Error Rate and Bit Rate limits

There are several factors in work in the quantum cryptography system that will considerable decrease the bit-rate from Alice, prior to the final agreement between Alice and Bob on a perfectly secret shared key. The factors restraining the bit-rate of the transmission is are; losses during propagation, the detector efficiency, and the weak pulses (intensity μ) introduced by Alice's laser to ensure that at maximum one photon at a time is coded with the same information. Not to forget of course, all eavesdropping attempts introduce errors. The detection process also introduces errors.

The collected rate of all errors e, is called the quantum bit error rate (QBER). If η is the detector quantum efficiency, μ the average number of photons per pulse, α the channel (fiber) attenuation coefficient, and L the transmission length, then the raw bit-rate R for Bob is

$$R = \eta\mu e^{-\alpha L} B\,.$$

where B is the bit-rate on Alice's side.

However, the key distribution is limited not so much to the bit-rate itself, but rather to the error-rate, $QBER$, for which an upper bound exist above where not all errors can be corrected without decreasing the bit-rate to zero. For perfect visibility the $QBER$, e, is given by

$$e = \frac{P_d}{P_d + e^{-\alpha L}\eta\mu}\,,$$

where P_d is the dark count probability for the detector.

When Bob have received and decoded the transmitted information he will end up with a number of raw bits. Depending on the chosen coding-scheme, a constant factor (on average) of bits will be lost at the demodulation, in our case about 1/2. These sifted bits will still contain errors occurring with the same error-rate and has to be corrected. Also, the raw-key is not perfectly secure and has to be compressed by privacy amplification to gain security. All these factors are the collective outcomes of the work of the key-distillation protocol. As this is the main-topic discussed in the paper, we will in this section concentrate on and investigate the decrease in bit-rate from Bob having the raw-bits to the safe and finally shared key between Alice and Bob. The decrease in bit-rate over the PQC will be discussed in section 6.3.3.

As a performance measure on the protocols, we use the so-called *effective bit-rate*, i.e. the rate in that gives the bit-rate decrease due to sifting, error correction, and privacy amplification. This effective bit-rate is given by

$$R_{eff}^{LB} = (1 - r_s)(1 - r_{ec})(1 - r_{pa})R\,,$$

where R is the raw bit-rate. This is a lower bound on the achievable bit-rate. We will derive R_{eff} for all protocols as a function of the error-rate e. For a specific value on

the error-rate R_{eff} will reach zero. This is the maximum error-rate e_{lim} when all errors are corrected. The fraction of bits abandoned in the sifting step is

$$r_s = \frac{1}{2}.$$

The fraction of bits lost in error correction r_{ec} is repeated here for all four methods;

$$r_{ec}^{coding} = \log_2(1+2\sqrt{e(1-e)}),$$

$$r_{ec}^{discard} = \frac{7}{2}e - e\log_2 e,$$

$$r_{ec}^{correct} = r_{ec}^{cascade} = 0.$$

The compression-size in privacy amplification does not only depend on the information leaked during reconciliation, but also on the information Eve obtained by wiretapping the quantum channel. A cruder estimate (see section 3.4) on the latter information t is given by

$$t = \log_2(1+4e-4e^2).$$

In this theoretical discussion we do not use any extra security compression for privacy amplification, hence g=0. This implies that Eve knows at maximum one bit of the final key shared between Alice and Bob. The compression is given by r_{pa}=t+q+g, where q is the fraction of bits leaked during error-correction. For the different methods we have,

$$r_{pa}^{coding} = \frac{1}{1-\log_2(1+2\sqrt{e(1-e)})} - 1 + \log_2(1+4e-4e^2),$$

$$r_{pa}^{discard} = \log_2(1+4e-4e^2),$$

$$r_{pa}^{correct} = \frac{7}{2}e - e\log_2 e + \log_2(1+4e-4e^2),$$

$$r_{pa}^{cascade} = 2 + \frac{1-(1-2e)^{0.73/e}}{2}\log_2\left(\frac{0.73}{e}\right)$$
$$+\left(0.73 - \frac{1-(1-2e)^{0.73/e}}{2}\right)\cdot\log_2\left(\frac{0.73}{e}\right)\cdot\sum_{l=2}^{w}2^{1-l} + \log_2(1+4e-4e^2).$$

We will now look at some plots comparing different effective bit-rates R_{eff}. R is here normalized to 1 for easy comparison with the channel capacity

$$C = (1-r_s)(1-H_b(e)) = \frac{1}{2}(1+e\log_2 e + (1-e)\log_2(1-e)).$$

All plots also show an upper bound, i.e. the best acheivable performance for all protocols. This is based on the minimum number of bits required to be exchanged over the PC in order to correct all errors. This was discussed in section 4.3.

$$R_{eff}^{UB} = \left(\frac{N_{\min}}{N_s} + t \right) R = (-e \log_2 e - (1-e)\log_2(1-e) + \log_2(1+4e-4e^2))R\,.$$

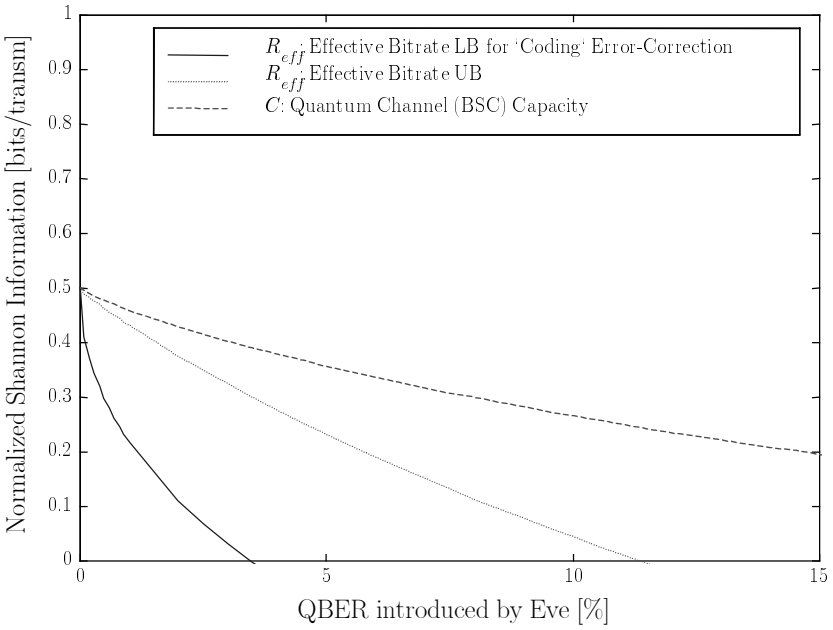


Figure 4.9: The *coding* protocol system performance. The y-axis shows the preserved fraction of bits of the normalized bit-rate R, achieved by Bob's after demodulation. The solid line shows gives a lower bound (LB) for the error-correction limit, $e_{lim} \approx 3.5\%$. The dotted line shows the best performance achievable and the dashed line the channel capacity.

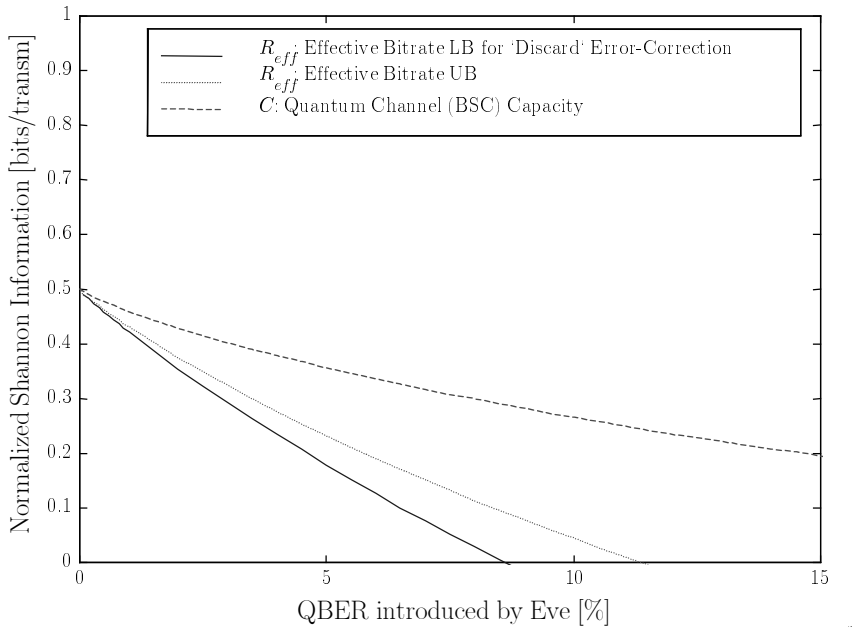


Figure 4.10: The *discard* protocol system performance. The y-axis shows the preserved fraction of bits of the normalized bit-rate R, achieved by Bob's after demodulation. The solid line shows gives a lower bound (LB) for the error-correction limit, $e_{lim} \approx 8.5\%$. The dotted line shows the best performance achievable and the dashed line the channel capacity.

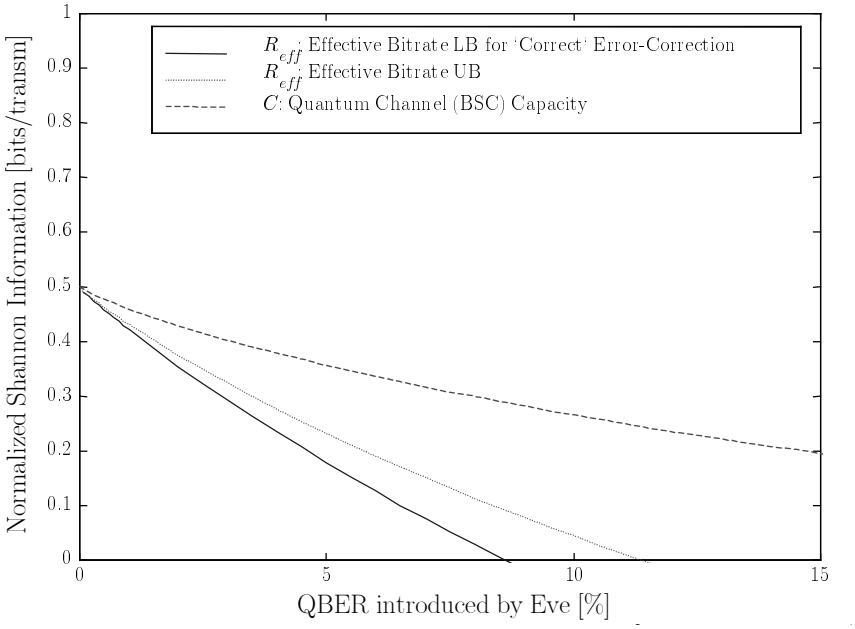


Figure 4.11: The *correct* protocol system performance. The y-axis shows the preserved fraction of bits of the normalized bit-rate R, achieved by Bob's after demodulation. The solid line shows gives a

lower bound (LB) for the error-correction limit, $e_{lim} \approx 8.5\%$. The dotted line shows the best performance achievable and the dashed line the channel capacity.

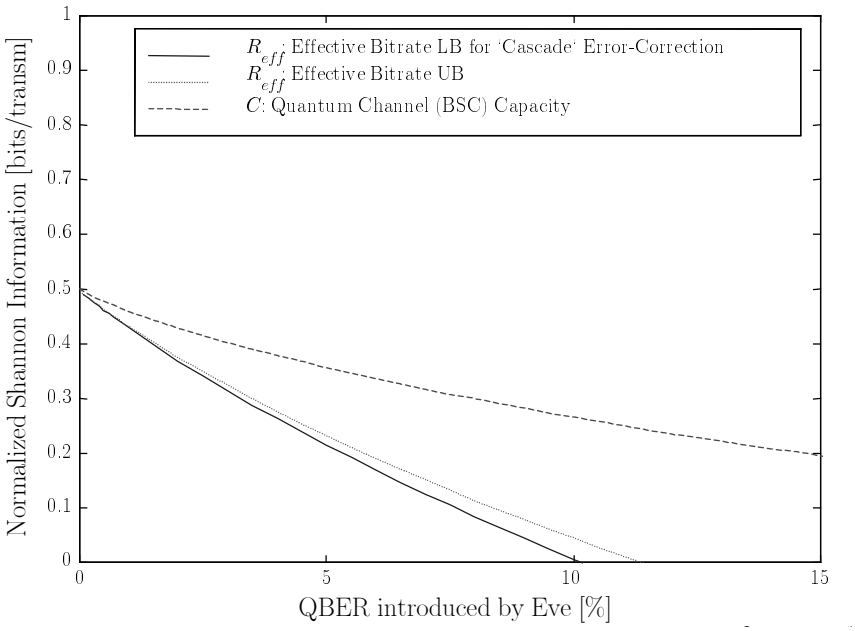


Figure 4.12: The *cascade* protocol system performance. The y-axis shows the preserved fraction of bits of the normalized bit-rate R, achieved by Bob's after demodulation. The solid line shows gives a lower bound (LB) for the error-correction limit, $e_{lim} \approx 10.2\%$. The dotted line shows the best performance achievable and the dashed line the channel capacity.

It is clear from the graphs that the best protocol at present is the *cascade* by Bennet and Salvail. Simulation-results using any of the above-examined protocols give a graph lying in between the lower and upper bound. This will be verified by simulation in section 6.3, except for coding, which is not implemented.

One of the conclusions from last section discussing privacy amplification was that, indeed, it more efficient to let information be lost to Eve during reconciliation because this will be compensated by the hash-function compression and the advantage of being able to choose a protocol not reducing the key-length. Cascade also works close to the Shannon limit, therefore this is a good approach.

To get a grip of the difference between removing errors and correcting errors vs. doing more or less compression, compare Figure 4.13 and Figure 4.14. The final effective bit-rate will be higher for cascade.

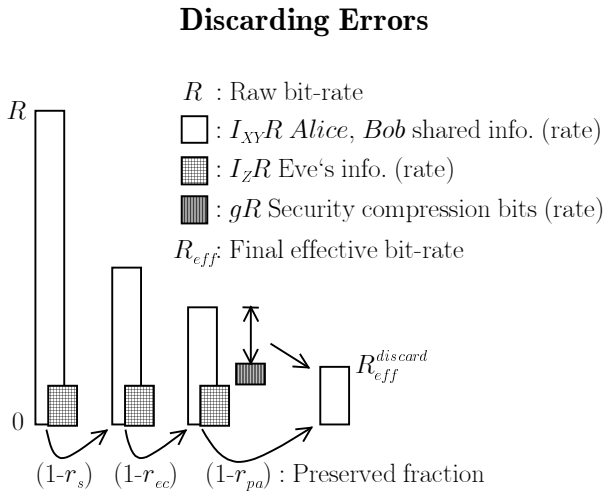


Figure 4.13: The effective bit-rate will decrease to only a small fraction R_{eff} of what is received by Bob at detection R. Extra security compression will reduce Eve's knowledge to an arbitrarily small value. Here $q = 0$.

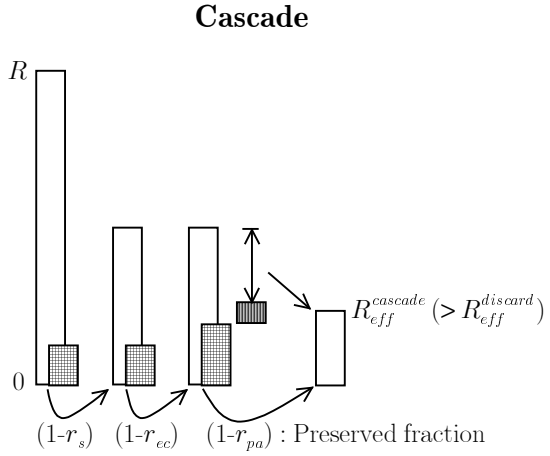


Figure 4.14: Compared to discarding errors a fraction of bits will be lost during reconciliation, $q \neq 0$, but $r_{ec} = 0$. Cascade will end up with a larger fraction of bits preserved.

4.6 Security and Eavesdropper Information Restraints

A common feature of the key distribution scheme reviewed in Chapter 3 is that the eavesdropper Eve cannot learn anything about the key without introducing errors among the sifted bits. We have learned from this chapter that the reconciliation protocols has an upper limit on the error-rate for which all errors can be corrected, and if we also want the effective bit-rate to be separated from zero, this limit is even lower. Thus, the privacy amplification step will work as an automatic test for eavesdropping. If Eve introduce too many errors Alice and Bob will end up with no key at all; the whole remaining part of the key is compressed. When the error-rate is above the some specific limit, based on the quantum channel error-rate plus statistical fluctuations, the conclusion that Eve is penetrating the line is far from irrelevant. An attack is detected with high probability.

Because of properties of privacy amplification it does not matter, however, from where the errors arise as long as Alice and Bob can agree over a key that is long enough for the application. The security can always be assured to exceed a specific value if the bit-rate is above zero, $I_E < 2^{-gN_s}/\ln 2$. Maintaining a longer bit-sequence would in practice, anyhow require a lower quantum bit error-rate.

Yet, in a more realistic condition the errors result from channel and detector imperfections even without eavesdropping. At the same time, the enemy can improve the quality of the channel, limited only by the law of physics to hide the evidence of tampering. Therefore, it is unsafe to ascribe errors as by natural causes, rather, all errors should be treated as if they arise from a malicious eavesdropper.

Some fundamental requirements can be stated about the security of a quantum cryptography distribution. If e errors are found in the sifted key of length N_s, then, after error correction revealing minimum N_{min} bits to Eve, a new key of N_f bits can be distilled by privacy amplification on which Eve, with a probability p_{safe}, has less than I_E^{tol} bits of Shannon information. The value of I_E^{tol} has to be in compliance with the security requirements for the actual application in mind.

The step now is to combine I_E^{tol} and p_{safe} to set a value for the security parameter g. If I_E is the information available to Eve on the final key, then, by making it satisfy $I_E > (1-p_{safe})I_E^{tol}$, we have put an restraint on I_E so that I_E^{tol}, guaranteed with a probability of p_{safe}, will not be exceeded [17]. Continued, we have

$$I_E < 2^{-gN_e}/\ln 2\,,$$

and so

$$g > \frac{-\log_2((1-p_{safe})I_E^{tol}\ln 2)}{N_e}\,.$$

By setting I_E^{tol} and choosing g from this formula, we can with probability p_{safe} obtain a key that is secure up to an arbitrarily level. Both p_{safe} and I_E^{tol} have to be agreed upon by Alice and Bob before distribution of any key. These two parameter are ready to be set in the interface of the simulation program of chapter Chapter 6. With I_E^{tol}, p_{safe}, and g, plus an estimate of t, Alice and Bob can now fix a value for the final compression, s=t+g. If all

parameters are set according to these directives, before any actual key distribution, the security requirements will be met.

However, there are no conditions under this set up that tells Alice and Bob how large their key will be constructed. One can gladly put in a restraint of this kind as well, thereby setting a minimum limit on the reconciled key-length. Before key-distribution they then have to calculate for a minimum value of the number of bits send from Alice, providing this desired final length with a certain probability and with all other security requirements satisfied.

Chapter 5

Experimental Quantum Cryptography

5.1 The Quantum Cryptography System at ELE

Quantum Cryptography has for long now entered the experimental ground. The first successful transmission of quantum bits took place 1989 in a setup by Bennet and Brassard et al. [3]. The experimental quantum cryptography system build at the Department of Electronics, KTH, is a so-called "Plug and Play" setup [20]. The name (borrowed from computer industry) is chosen because the system parts are very easy to connect. It is an auto-balanced QC-setup based on an interferometer with faraday-mirrors that does not need any alignment. The present system uses phase encoding of the B92 protocol. Phase encoding is preferred over polarization encoding, because the phase of a photon is better preserved than the polarization in telecom fibers. The birefringence of fibers and the effect of the environment makes polarization fluctuate more randomly.

All realization using phase coding schemes are based on interferometrical setups, using a Mach-Zehnder interferometer at both Alice and Bob, or a Michelson interferometer with one long arm going to Alice. The latter is used in the current implementation. Several problems are solved by using this variant of an Michelson interferometer, an inconvenience using two Mach-Zhender interferometers is that

they have to be adjusted to each other in order to obtain good interference, and they must have the same path lengths. This is not the case, using only one interferometer.

We will now turn to an explanation of the setup, see figure Figure 5.1. The principle is to get interference at Bob between two weak pulses. Bob and Alice can change the phase of these pulses. If this interference is constructive or not can be detected, thus Alice's and Bob's independent choices of the phase can carry information from Alice to Bob. Bob starts by splitting the sending pulse into two parts, one goes to Alice and returns phase-modulated (coded), the other one is reflected by Bob's apparatus and phase-modulated as well. If these two pulses undergo constructive interfere at Bob a pulse is detected, otherwise not. In this way, Bob will get a detection if he chooses the right phase. The pulse going back from Alice carries her secret information and it is important that this information about one bit is transported by only one single photon. Alice has apparatus to assure this. It is difficult to generate single photons, therefore existing schemes relies in weak pulses instead. These weak pulses will have an intensity of μ=0.1 photons per transmission. The probability that two or more photons are sent will then be given by $\mu^2/2$ (a laser exhibits a Poissonian distribution of μ). This will of course decrease the bit-rate performance for the system, since on the average only every 10:th (μ^{-1}) photon will arrive at Bob.

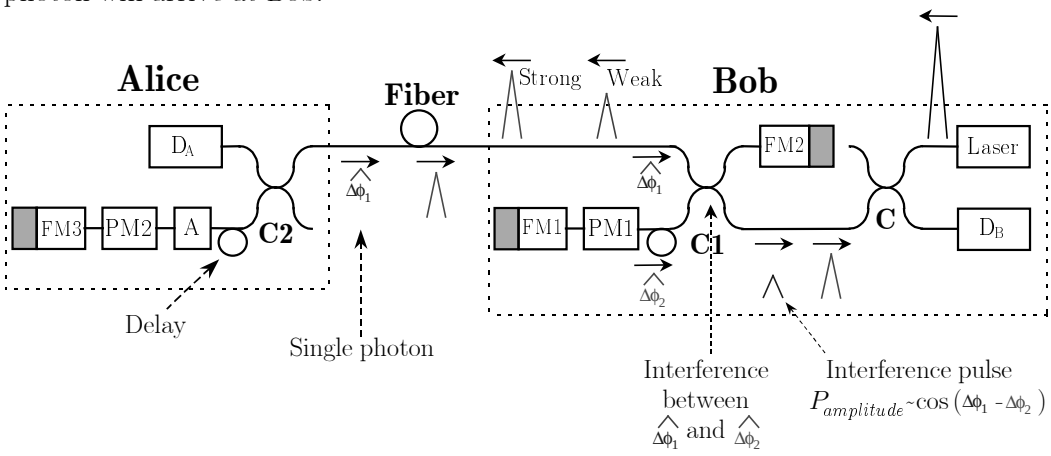


Figure 5.1: 'Plug and Play' system at Department of Electronics, KTH, using phase-encoding in the BB92 scheme. The laser-pulse from Bob is split into two pulses, one comes back from Alice modulated and the other one is reflected at Bob and also modulated. Both pulses will interfere at Bob and can carry information about a 1 or a 0 in either one of the bases, by Alice's choice of modulation.

The procedure can be described by these main steps:

- Bob sends a pulse from the laser.
- At C1 (fiber coupler), the pulse is split. The weaker went through Faraday Mirrors FM1-FM2 then onto the fiber, the stronger went directly onto the fiber.
- When part of the stronger pulse (split in C2) reaches detector D_A (PIN), the detector triggers Alice's phase modulator PM2, which puts a phase to the weaker pulse.
- Both pulses are reflected at Faraday mirror FM3, and attenuated by attenuator A to the single photon level.

- When the pulses again reaches C1, one part of the strong pulse goes via FM2-Bobs Phase Modulator PM1, where it acquires a phase shift.
- The weak pulse, with a phase shift of Alice, interfere with the strong pulse, with a phase shift of Bob.
- For constructive interference, Bob (who has the timing information since he sends the initial pulse) can gate his single photon detector, D_B to select only pulses with the right timing information.

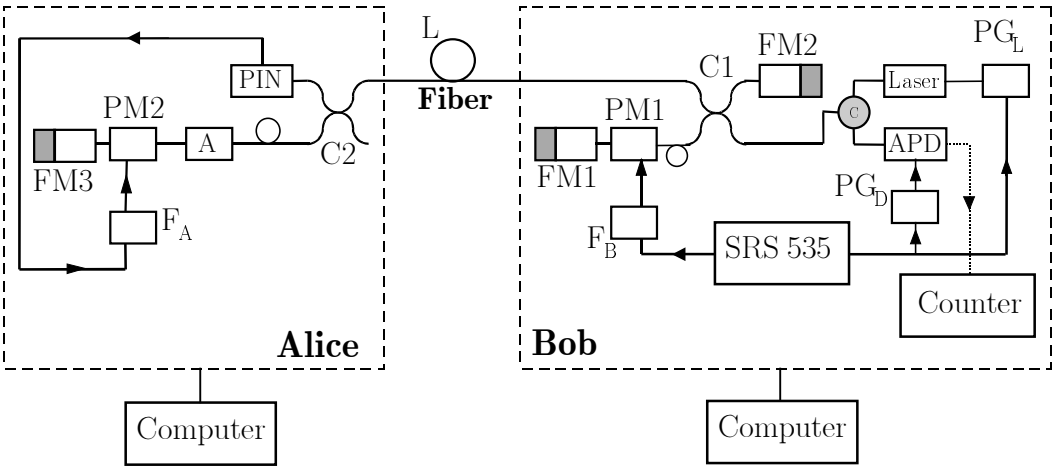


Figure 5.2: 'Plug and Play' system at Department of Electronics, KTH, using phase-encoding of the BB92 scheme. The SRS535 is the (master) delay pulse generator, F_A and F_B are function generators. PG_D and PG_L are pulse generators. APD is the Single Photon Detector (Avalanche Photo Diode). A is an attenuator.

Figure 5.2 shows a more detailed block-diagram of the current setup. Some instrument equipments are shown, as well as the two computers, soon to be connected to the system in order to automatically and fully control the key distribution setup. These computers are in turn controlled by a Labview program, and Matlab software. All eventual tunings of the system and both modulators will be controlled by the computers. The computers will also be connected to each other by a public channel in form of null modem, TCP/IP or such.

5.2 Experimental Results and Challenges

As all classical communication system has problems with noise in transmission, so do the quantum cryptography system. A system is never stronger than its weakest link. The QC-system performance is limited by the quality of its components. Noise arises during generation, transmission and detection of the quantum states. There will not be a full correlation between sent signals and received signals.

For a cryptography-system where the basic assurance of security is based on the ability of detecting an eavesdropper, this introduces a problem. The adversary is detected by her introduction of noise, but how can this noise be distinguished from transmission noise? It can't. It is therefore necessary to see all noise as coming from an eavesdroppers activity. In Chapter 4 we learned that for error-rates above a

specific limit, the key will be compressed into nothing, and communication is terminated.

To this date, QC has been experimentally proven feasible. The most important technological challenge is the development into better photon counters, whose noise limits the practical transmission distance. The detector occasionally register a photon even when no photon is present to trigger an avalanche in the APD, and sometimes fail to register a photon. Wrong results will be the consequence. The main technical limitations is the performance of the detector in terms of quantum efficiency, dark counts, and time resolution.

Somewhat successful development towards better detectors have been reported from ELE at KTH; the system operating at long wavelength telecom window (1550nm) uses gated InGaAs APD:s as single photon detector. The error-rate is presently 3% for an transmission-distance of 10km using an intensity of μ=0.1, see Figure 5.3. The detector is currently liquid nitrogen cooled and gives a quantum efficiency of η=18% and a dark count probability $R_{dark}\delta T = 2 \cdot 10^{-4}$, where R_{dark} is the dark count rate and δT=5ns is the gate width. By further optimizations, like improving detection electronics and using better cooling system, a transmission distance of L=100km would not seem unfeasible. For a more thoroughly report on this system setup see [15] and [8].

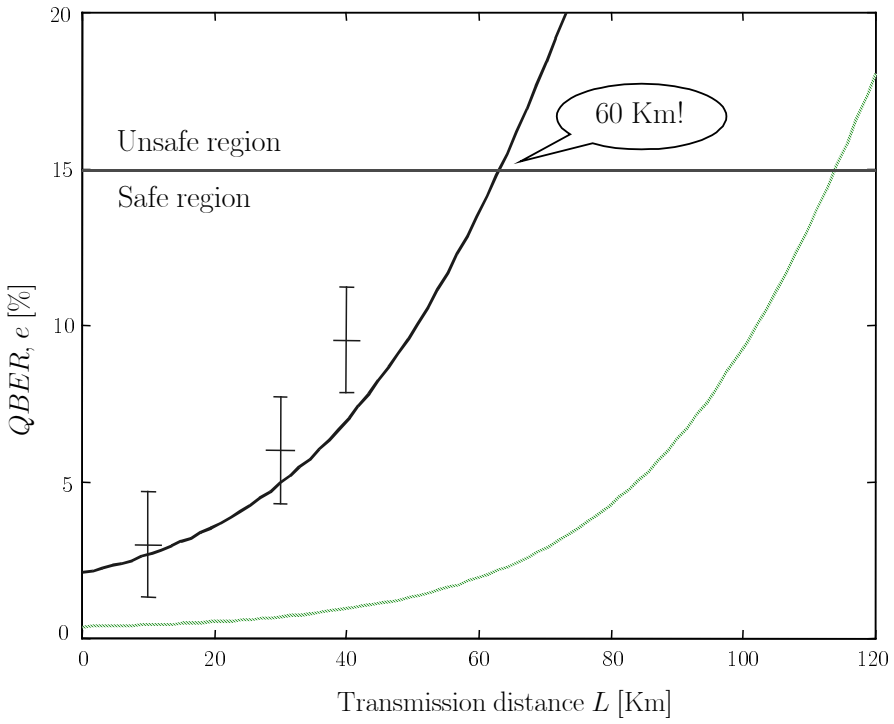


Figure 5.3: The ELE-system performance. Experimental results are obtained for 10, 30 and 40 km of fiber lengths. The distance is limited to 60km, then to many errors are introduced for safe communication, if the limit is set at 15% (depending on the information leakage estimate used). The dotted line shows performance in future, possibly with 10 times detector improvements.

Chapter 6

Interface for Quantum Cryptography System

6.1 Introduction

In order to control the experimental system setup and being able to distribute a key we need some software implementations. The software will implement and simulate all steps; from the sending of the random bit-string over the fiber to the sifting, error correction, and privacy amplification, all according to the quantum cryptography scheme described earlier. We have two different kinds of software that communicate with each other and the experimental setup. The first one is fully implemented presently and will be treated in the following sections.

The purpose of the main software-program part should principally be to make up a graphical user interface for both Alice and Bob were all necessary parameters are set and the resulting key is presented. In the background, the program should utilize the reconciliation (error correction) protocol and the privacy amplification steps to distill the final secretly shared key. Notice that we in this first system only use one computer hosting both Alice and Bob, but no communication in between Alice's and Bob's software are allowed outside the strict rules of the protocol. This part we hereafter refer to as the software implementation. The code is written in MATLAB, that allows easy simulations.

The second part of the interface software is what we refer to as the software-to-hardware interface. In other words, the main purpose of this code is to control all the lab-instruments that, in its turn, are controlling the entire system setup. To accomplish this we will use Labview, an instrumentation-program that uses a data-acquisition board, (DAQ) on the computer to give order to instruments and receive data from the same. This part of the software-implementation is controlled by higher orders from the main program.

The abstraction level is set at the distribution of the raw key. A random data-sequence is generated by the labview-software in Alice's instrumentation, and sent over the private channel to be received by Bob's instrumentation. Then, by the protocol in the main program using the raw data on each Alice's and Bob's side, the final key will be distilled making use of the sifting, reconciliation, and privacy amplification parts.

Another feature that is implemented in the software is the ability to simulate the quantum channel. A simulated private channel replaces the software-to-hardware program and the fiber channel setup. By option, you can also choose to simulate different strategies of the eavesdropper Eve. You will be able to perform simulations testing the performance of different protocols, e.g. the actual bit creation rate against different tolerated levels on Eve's final information or the bit-rate versus $QBER$, e, introduced by Eve.

The objective is to supply a user-friendly interaction utility with the quantum cryptography system setup in the lab. To have a program that lets you analyze the performance of protocols, and to give implications of how efficient future error-correction methods can be implemented. An important parameter to investigate is the upper limit of the quantum bit error rate introduced by Eve, still being able to distill a secure key. We also look for a bound on the total bit-rate R for a known $QBER$, i.e. by what fraction do we shorten our key during reconciliation and privacy amplification. As long as the $QBER$ remains below the limit we can actually forget about the eavesdropping taking place and still receive a secure key, albeit with a reduced bit-rate. Furthermore, we want an estimate of the total information Eve has learned after completing all steps, compared to Alice's and Bob's mutual information. How good is the security of the final key for different eavesdropping attempts? For practical purposes, it is relevant to restrict the eavesdropping to individual attacks, and therefore only the cases of *intercept/resend* and *beamsplitting* are investigated.

6.2 Software Implementation

In the experimental system setup the information is modulated using the B92 scheme, hence the practical protocol implemented in the software also works within the frames of this scheme. You could say that the software implementation is just a realization of the public channel, used to correct and make the key secure after it has been transmitted over the private channel. Private and public channels were discussed in section 3.3 and 2.4.

The program embodies a graphical-user-interface (GUI) function that serves as a command window. This function `qcinterface.m` only handles parameter-choices and result-graphics and therefore, in its turn, calls a sub-function `main.m` that realizes the protocol. Two other graphical windows can be opened which represents each Alice's and Bob's side. A sample of the key for each side is shown here and their development through the protocol - towards a commonly shared key - can be followed. No communication whatsoever between Alice and Bob is allowed except

within the rules of the protocol, although it could be tempting to cheat as they reside on the same computer. This distinction is maintained because of desired future possibilities of putting Alice and Bob on separate computers. Then for the public channel, you will just have to integrate the program with some reliable error free communication-channel, like a zero-modem, telephone-modem, Ethernet, Internet protocol (TCP/IP), or such.

The steps performed are in order; the random bits generation, Alice's transmission of the bits, Bob's reception of the bits, sifting, reconciliation (error correction), and last privacy amplification. These steps were treated in Chapter 4. Also, see Figure 4.1.

As discussed, the program start off with the input of the raw-bits, either received from LABVIEW, i.e. a real transmission in the quantum channel, or from a simulation. The bit-sequence generated by Alice comes either from truly random bits generated by an external random bit generator controlled by Labview, or for small simulations, from a pseudo-random generation in Matlab. (Preparations have been made for future implementations of the external device. It is not used for the moment). Now Alice and Bob have one version each of the raw-bits, but they are not the same. As covered in Chapter 3 the raw-bits are now undertaken sifting, that is, Bob supplies Alice with knowledge of the bases used measuring the bases on Bob's side. Alice compares all bases with hers and sends information back to Bob telling which qubits were measured correctly. Now they have around 50% of the bits left, eventually containing errors.

The steps remaining are; error correction, to reconcile the key, and privacy amplification, to make the key more secret and decrease Eve's information to an acceptable low level. How these principal steps are practically implemented will be discussed in the following sections. See also Figure 6.1.

In the GUI-window, two important parameters are set, concerning final security (I_E^{tol} and p_{safe}). I_E^{tol} is the maximum tolerated level of the Shannon information for Eve on the final key. It should be chosen very small, in the order of at most 10^{-10}. p_{safe} tells us with what probability this value will be exceeded, also to be very small, in the order of at most 10^{-10}.

For the program to work in practice you need to do some test on the actual quantum bit error rate. If you want the program to work as realistic as possible, these tests should be chosen by marking a box in the GUI before execution. The first test, $ESTIMQBER$ (before any secure key-transmission is made), is to test the quantum bit error rate of the channel, that is, e.g. the detector efficiency. The other test, $TESTQBER$ (before error correction), is to estimate the actual $QBER$. This estimate is used to set the size of the block-length and other parameters for the reconciliation-protocols. In case these tests are not chosen, the program will work anyway, with values set in the case of a quantum channel simulation. All parameters are explained more in section 6.2.2 and 6.2.3.

6.2.1 Program Structure

Within the frames of the protocol, some information will be exchanged between Alice and Bob. The program is designed for Bob to act as the ruling part. In other words, Bob is the one who takes action and calls for information from Alice when needed, in order to proceed with the protocol. This maybe in the opposite way of the nomenclature presently used in the literature community, but is nevertheless efficient in an implementation point of view, and it makes no difference in practical use, which direction the information travels.

A model of the program, i.e. the program structure diagram, can be found attached in Appendix B. The GUI-function `qcinterface.m` calls `main.m` who is the 'boss-function', designed to be on either side of Alice and Bob. Presently they reside at the same computer, in future they can be separated with meager modification of the code. The functions of Alice and Bob are implemented in `alice.m` respective `bob.m`. Bob has routines `bob_protocol_xxxxxxx.m` and `bob_graphics.m`, that handles the protocol and the graphics respectively, i.e. prints the key in a separate window. `xxxxxxx` stands for the name of either one of the three different types of protocols implemented; `cascade`, `correct`, `discard`. The same routines exist for Alice. These functions are called for using an argument like `method` or `service`, in order to make each function perform different tasks (a kind of pseudo object oriented class-programming.) The names of the different methods in the program model (Appendix B) are chosen to provide a hint of their function.

The public channel is implemented in `publicch.m` and the private channel (for simulations) in `quantumch.m`. When using the real quantum channel, i.e. fiber, `quantumch.m` calls for another function `labview_quantumch.m` that establishes the connection to the experimental setup. At this stage, all that `publicch.m` does is to forward all order from Bob to Alice and vice versa. When separating on two computers later, this should be the only function to change, by adding code for Ethernet-connection or such.

Main Steps in Program Code:

1: Start: `qcinterface.m` calls `main.m`, that, in it's turn calls `alice.m`

2: Alice initiates bits: `alice.m` calls `quantumch.m` were the random bits are generated; these are send over quantum channel.

3: Bob receives: `main.m` calls `bob.m`, that, in turn calls `quantumch.m` and `bob.m` receives the bits.

4: Protocol: bob.m calls e.g. `bob_protocol_cascade.m` that in turn calls `alice_protocol_cascade.m` and the protocol connection is so established. The following steps are performed:

i. Sifting (compare bases.)

ii. Reconciliation (error correction.)

iii. Privacy amplification.

5: End: `bob.m` sends his key back to `main.m`. `main.m` also receives key from `alice.m`.

Table 6.1: Main steps in the Quantum Cryptography Interface; a basic state-flow, describing the connection between the routines making up the program. The lasts step 5) should of course **not** be done in a real implementation

suited for secure communication. This step is only for simulation reasons, so that Alice's and Bob's keys can be compared for errors.

6.2.2 Simulation of The Quantum Channel

In order to investigate the performance of different protocols, it would be very nice not being required to use the real quantum channel. Therefore, it is possible to simulate the channel and different eavesdropping attacks on it. The implementation is very simple. In `quantumch.m` is also implemented the polarization modulator and demodulator, that is the bits Alice send are structured together randomly with either choice of `rectangular` or `diagonal` polarization. This is saved as object `photons`. These `photons` can then be demodulated by Bob into bits, by choosing in the same way, either polarization-basis randomly - if he chooses the correct base he will receive the corresponding bit - if he chooses wrong base he will (not to his knowledge) receive a random bit. Eventually Eve could also have been eavesdropping and so changed the value of several bits, or changed the polarization base and introduced errors. This is implemented by simply flipping the bit-information in a photon with the probability set by a parameter. It covers the two individual-signal attacks; *intercept/resend* and *beamsplitt*.

The parameters used to realize the private channel are; the photon intensity (μ), the detector efficiency (η), the $QBER$ for channel and detector, and the $QBER$ introduced by Eve. The implementations of these parameters are straightforward; the photon intensity tells how many bits should be modulated at Alice's side and sent over the channel, the detector efficiency will be the probability that a photon is demodulated at all, and the channel $QBER$ are all the errors introduced by the detector at Bob's side by flipping his received bits with a probability set by a parameter value.

6.2.3 Graphical User Interface

To open the interface-window, run `qcinterface.m` in MATLAB. Your first choice is how many bits you want Alice to start off with, and send to Bob, e.g. 4000 bits. If you like to check the quantum channel for its quantum bit error rate you must also specify how many bits you want to use for the test. This gives an estimate of $e_{channel}$. By fraction, you can also choose how many bits before error correction that should be used to estimate the $QBER$, e, and so the block-lengths used in the protocol. Mark the box for *simulating quantum channel*, fill out corresponding parameters. You can choose what error correction protocol to use, and if you want to do privacy amplification.

There are two security-parameters; the maximum tolerated information, I_E^{tol}, of Eve on the final key, and the probability, p_{unsafe}=1-p_{safe}, that this value will actually be exceeded. Both can be chosen arbitrarily low at expense of the final key rate, as long as the $QBER$ is below a maximum value, discussed in section 4.5.

The value for parameter *Random noise seed* is used as a seed for all pseudo-random numbers generated in the program. It is thus possible to repeat a simulation with exactly the same outcome. If this parameter is set to '0' then a random seed will be used. Run a simulation by pressing 'Start Simulation'.

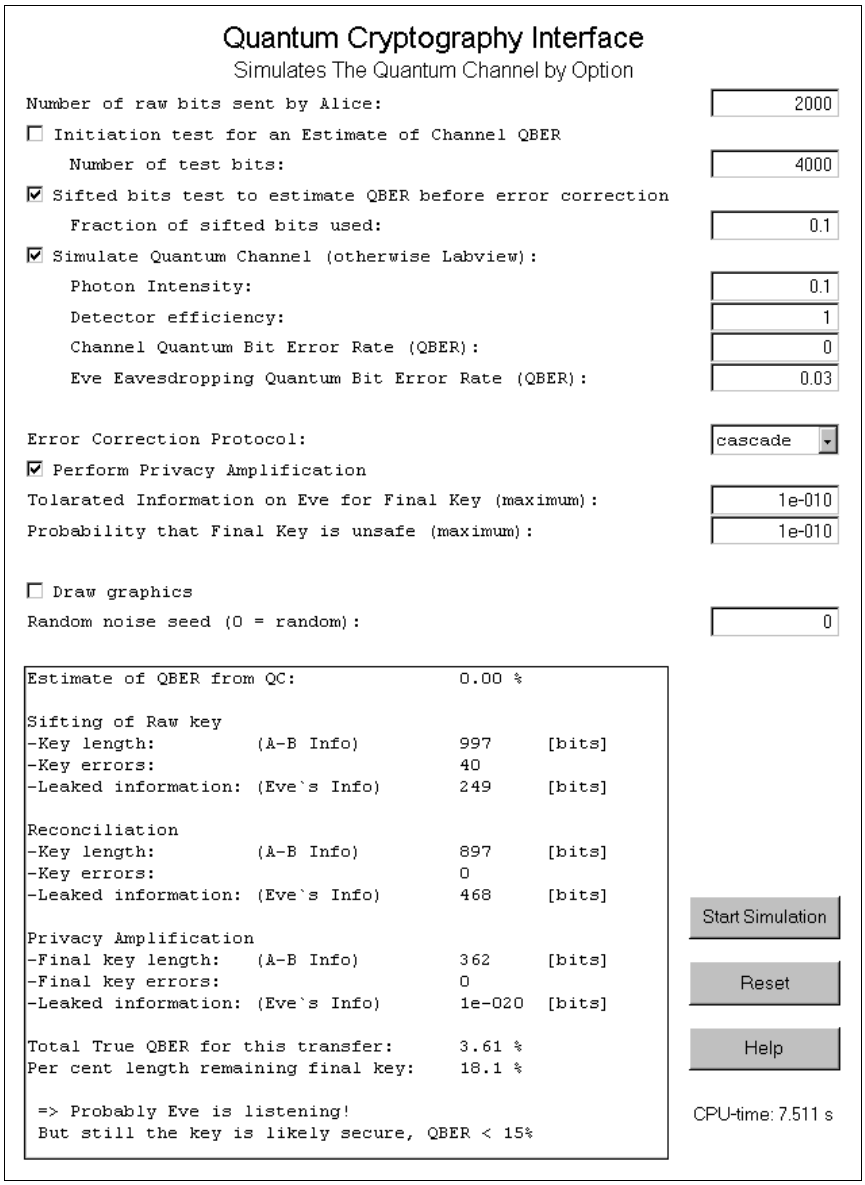


Figure 6.1: A sample MATLAB-GUI of the Quantum Cryptography Simulation Program. Result from one simulation appears in the lower-left box.

6.3 Simulations

Three of the reconciliation protocols presented in Chapter 4 are implemented; `xxx_protocol_discard.m`, `xxx_protocol.correct.m`, and `xxx_protocol_cascade.m`. *Discarding* errors and *Correcting* errors are implemented mainly for comparison reasons. *Cascade* is the most effective method, working close to the Shannon limit as discussed earlier. This is verified by simulations.

6.3.1 Error Correction Protocols

From section 3.4 we saw there exist different estimations for Eve's knowledge, a simpler and a cruder version. Figure 6.2 and Figure 6.3 shows simulations verifying the performance of the protocols *Discard* and *Cascade* for both estimates. The normalized information is calculated knowing the number of sifted bits and the final key length resulting from simulation for different error-rates; I_{AB}=N_f/N_s. The same applies for Eve's information I_E, but now we compare the estimate of her acquired number of bits with the sifted key length. The difference between the curves can be seen as the final secret information that can be extracted from the key after compression by privacy amplification. These plots thus show the state after error correction but before privacy amplification. For the *discard* protocol and with the cruder estimate we obtain a bound at 10.5%. The difference here compared to the theoretical 11.5% is due to the poor performance of this protocol. The same argument applies to the simple estimate giving 13.5% compared to 15%.

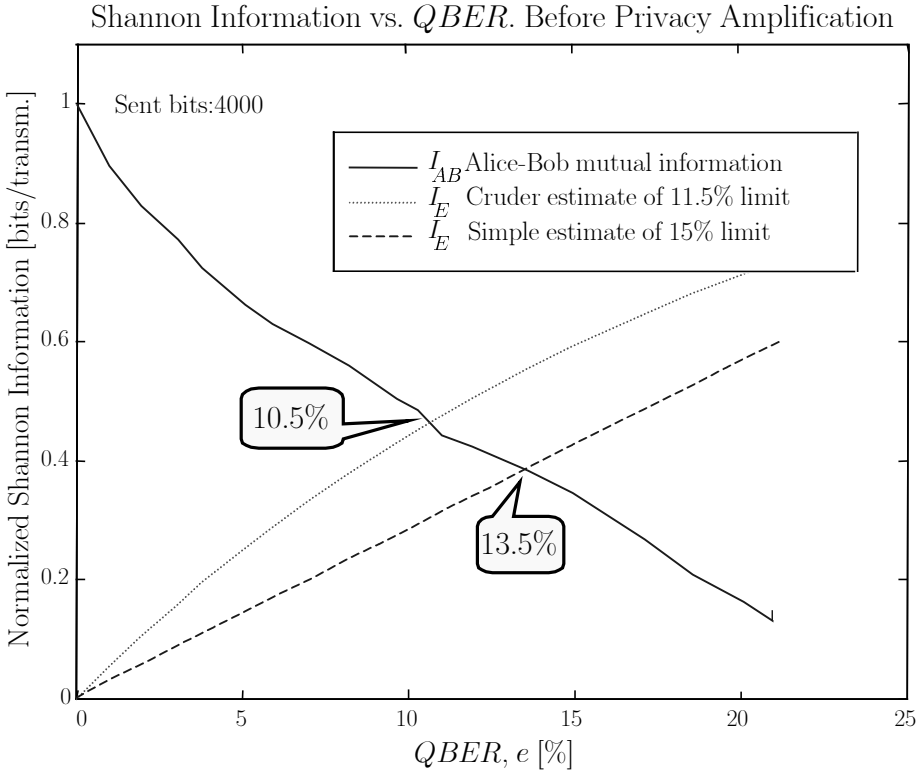


Figure 6.2: *Discard* protocol. The solid line showing Alice's and Bob's maximum mutual information for BB84. It is decreasing with w.r.t the error-rate. Eve's knowledge is consequently increasing. At some point the curves meet, and there's no secret information that can be extracted from their key. Two limits e_{lim}, are achieved, one for each estimate of Eve's information. These are lower than the theoretical (based on error-correction at the Shannon limit).

For the *cascade* protocol we instead find the corresponding limits 11.5% and 15%. The simulation does not give enough resolution to see that in fact these limits are close to, but not at, the theoretical limits.

The information, q, lost to Eve during error correction for the *cascade* protocol is calculated in the software implementation by counting the number of parity-checks.

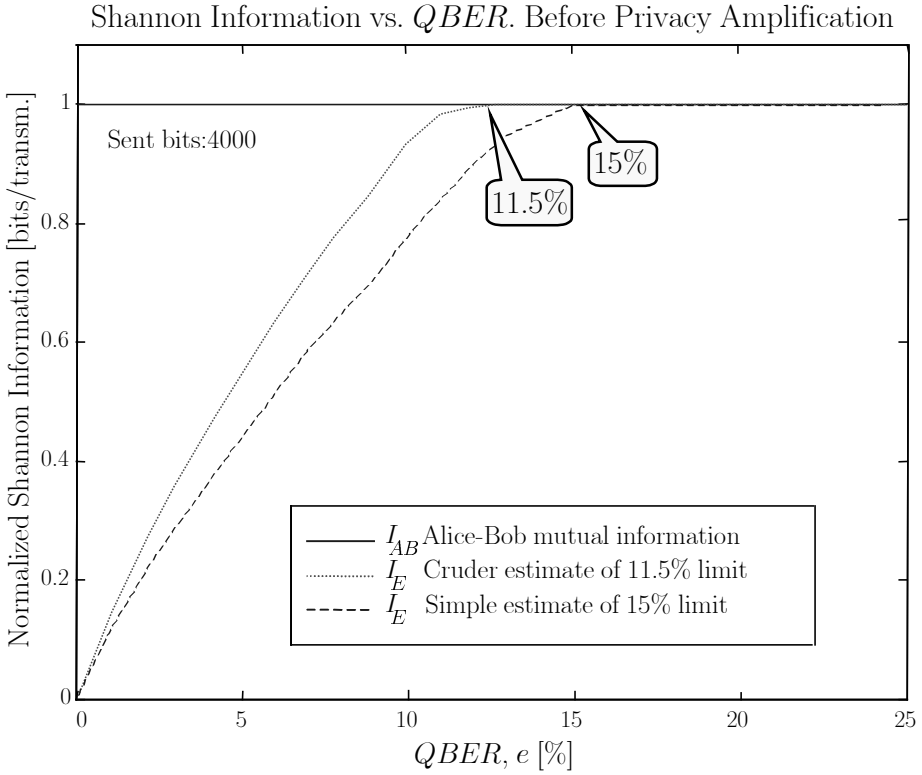


Figure 6.3: *Cascade* protocol. The solid line showing Alice's and Bob's mutual information; being constant because errors are corrected. Eve's knowledge is consequently increasing w.r.t the error-rate. At some point the curves meet, and there's no secret information that can be extracted from their key. Two limits e_{lim}, are achieved, one for each estimate of Eve's information. These are very close to the theoretical (based on error-correction at the Shannon limit).

6.3.2 Privacy Amplification Protocol

The privacy amplification step is implemented by use of a linear universal hash-function as stated in section 4.4. That is, a matrix is made consisting of randomly chosen 1:s and 0:s. Its size is $N_e \times N_f$. We make a modulo-2-wise multiplication between this hash-function and the reconciled key (length N_e). Out comes a compressed key, hopefully secure, and with length N_f. Parameter q is calculated by counting parity-checks minus the number of bits discarded, t is estimated by a formula based on the error-rate, and g is decided from the parameters set in the user-interface (I_E^{tol} and p_{safe}). The actual error-rate, e, is well known since it is easy to count how many errors have been corrected.

The effective bit-rate R_{eff}, (like calculated in section 4.5) is plotted in Figure 6.4 and Figure 6.5 (simple and crude estimate of t) for a simulation sending 4000 bits as the raw key and 15 simulations for each error-rate plotting value. The fluctuations

are large from each simulation to the other regarding the final key length. Therefore averaging is necessary. Also shown is the bit-rates with error-correction at the Shannon limit with and without security compression g. Note that for g>0 these simulated curves do not necessarily have to lie above the *lower* bound calculated for each protocol in section 4.5.

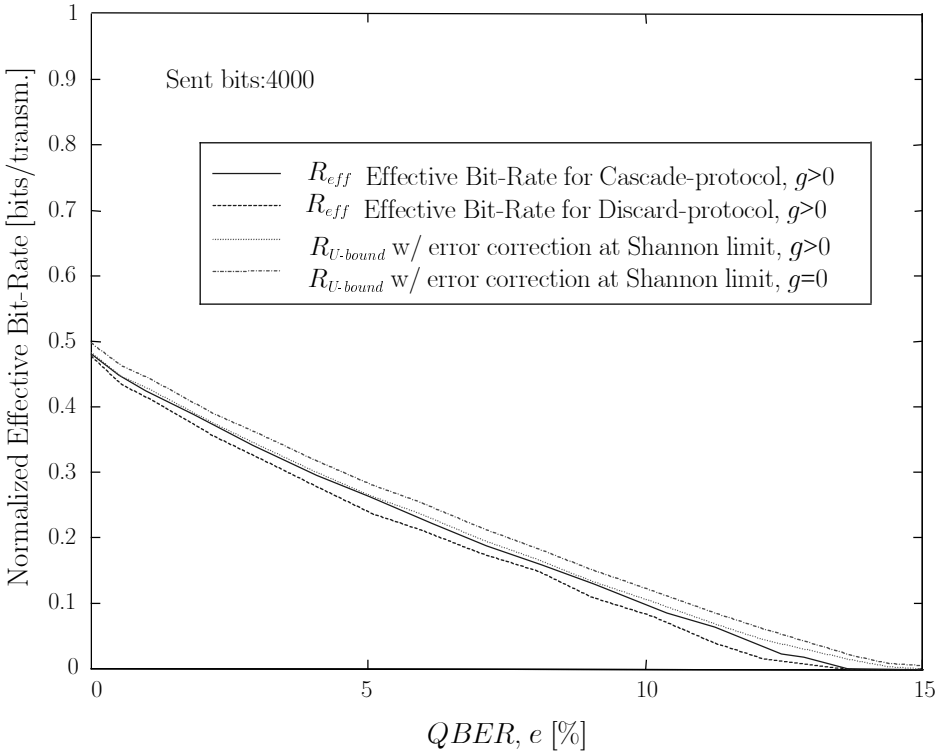


Figure 6.4: Using the *simple* estimate of t, simulations then give these effective bit-rates for *discard* and *cascade* protocol. The *cascade* protocol works very close to the upper bound for error correction at the Shannon limit (g>0). For error-rates larger than 12-13% you would in practice have to transmit bits for a long time to obtain any useful key-length.

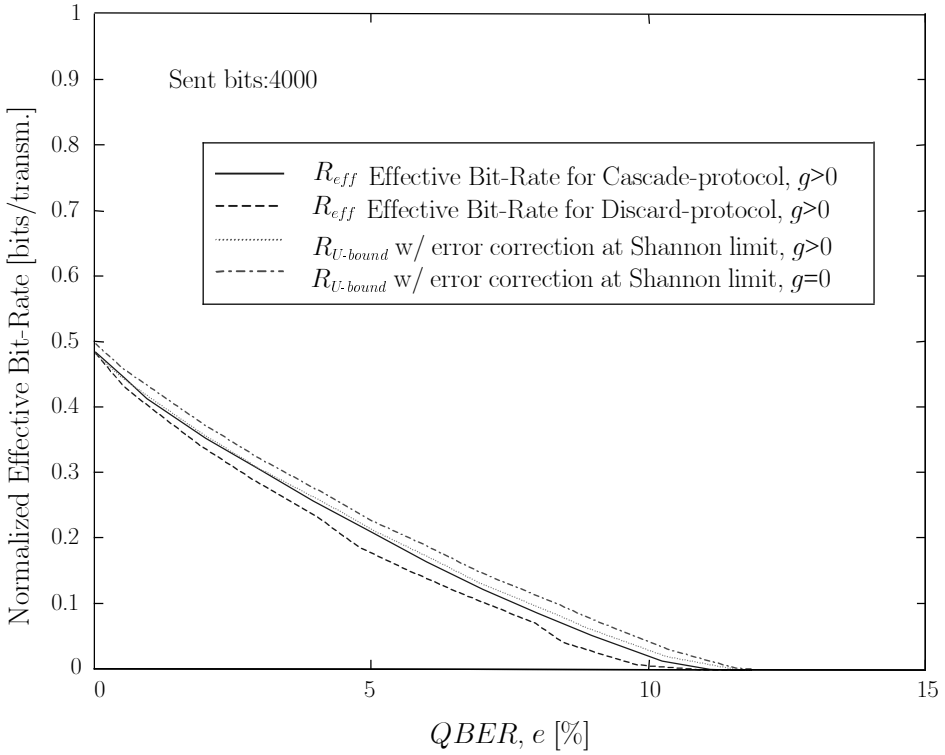


Figure 6.5: Using the *crude* estimate of t, simulations then give these effective bit-rates for *discard* and *cascade* protocol. The *cascade* protocol works very close to the upper bound for error correction at the Shannon limit (g>0). Compare also to Figure 4.9 and Figure 4.11 and the lower bound on the performance for the two protocols.

6.3.3 System Performance

We have now investigated the bit-rate decrease from the starting stage of the distribution process having the raw-bits, to the final stage of having a secure key. Plots showing this result were provided in the preceding section.

However, for all practical realization, more interestingly would be to also take into account the extensive decrease in bit-rate over the quantum channel. There are mainly three reasons for this bit-rate loss; first we have an intensity of only μ (typically μ=0.1) on the lightpulses. This means that on the average only μ (every 10:th) photons are sent from Alice for each transmission, to ensure that not more than one photon is sent at a time. Secondly, we have the fiber loss described by the fiber attenuation constant α, and the fiber length L. Only a fraction $e^{-\alpha L}$ will reach Bobs detector. As a third bottleneck we have the detection efficiency, η (the detector fail to register photons). These factors put restrictions to the bit-rate achievable by

$$R = r_f B = \eta\mu e^{-\alpha L} v \,,$$

where v frequency sent out by Alice (r_f is the fraction of bits lost in the PQC and B is the original bit-rate).

The problem about the photon intensity needs some additional comments. There is a tradeoff to be concerned with. Of course we would like μ as high as possible (equal to 1) but we cannot easily generate single photons, therefore we set the intensity to some lower value to assure that only very rarely more than one photon is send. This is important, remembering the eavesdropping strategy of beamsplitting. Eve will learn more information with higher μ. Unfortunately, this decreases the bit-rate of Alice and Bob and they get to share a lower amount of bits. There will always be some optimal value for μ where Eve learns a small amount of information from beamsplitting, at the same time as Alice and Bob have a fairly high bit-rate.

Norbert Lütkenhaus of Helsinki University, has recently investigated this problem in detail. By kind permission, we have borrowed his software, to make some predictions. The theoretical background and derivations are beyond the scope of this thesis. Since the work is not yet published, we will not reproduce any formulas, rather only present some results of interest.

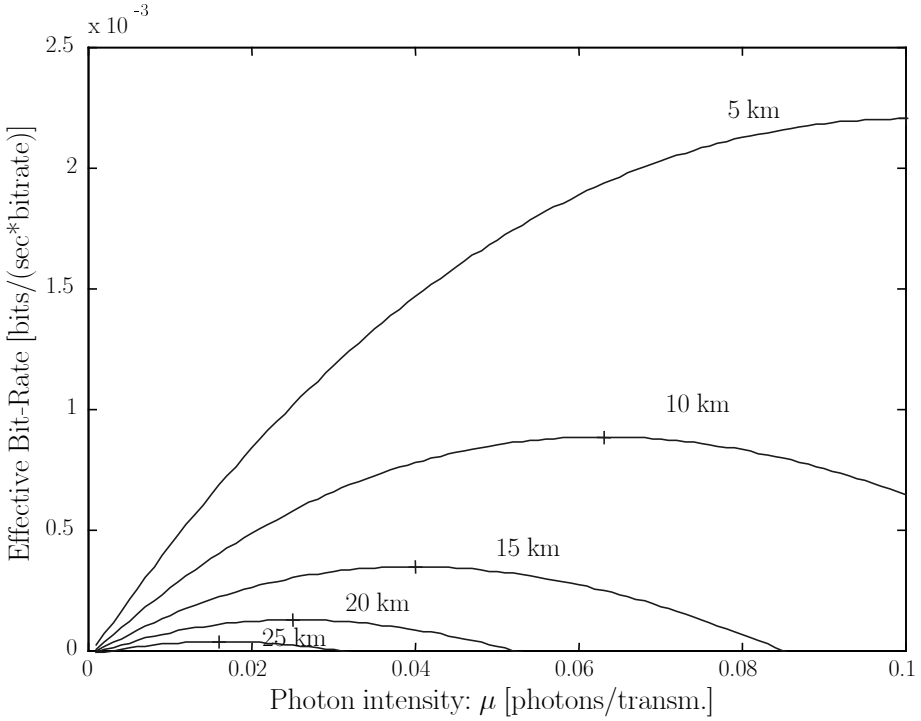


Figure 6.6: Effective bit-rate for the whole quantum system versus the photon intensity for different fiber-lengths [5 10 15 20 25 25] km, and with the *cascade* protocol in use. Parameter values: detector efficiency η=0.18, fiber loss constant α=0.38 [dB/km], dark count rate R_{dark}=10^{-5}. Alignment error between fibers = 0.008 plus an additional constant loss of 5dB. The curves give the optimal value for the photon intensity for each transmission distance. This is optimal in the sense that Eve is allowed a small as possible amount of photons to eavesdrop, at the same time as Bob will detect as many information-qubits as possible.

In these calculations more sophisticated attacks based on intercept/resend and beamsplitting are considered. Eve may be in possession of a fiber without attenuation, and, replacing the original fiber with this one, she can be in full control

over the intensity and the pulse rate received by Bob. Therefore, as Eve knows, for longer fiber-lengths, a larger amount of photons is attenuated. Eve can use her perfect fiber to take a larger fraction of photons to measure, and resend to Bob without any attenuation at all. In order for this not to happen, Alice has to keep down the intensity for longer lengths of the fiber. Results are plotted in Figure 6.6 for different lengths of the fiber. The photon intensity corresponding to the *maximum* (optimal) bit-rate for that specific fiber-length should always be chosen. We can see in the plot that the effective bit-rate is in the order of 10^{-3}. This means that if we require a bit-rate of 100 bits/sec (reasonable) then a frequency of $v = 100/10^{-3} = 10^{5} =$ 100 kHz is needed for the laser pulses in the transmission part of the quantum system.

Chapter 7

Conclusions

7.1 Experimental Quantum Cryptography

To this date, experimental quantum cryptography has been proven feasible in several labs all over the world. Experimental setups are currently able to generate sifted keys at an acceptable bit-rate and with low error-rates. The work behind this paper is a part of the next step; to investigate under what circumstances it is possible to implement practical protocols that from a sifted key can generate truly secret keys. This step requires public communication that is controlled by protocols.

The demand on protocols is to have high success-probability in reconciling the key and to produce keys at a high bit-rate. About the overall performance can be said that the most important task is to increase the bit-rate rather than to decrease the error-rate. The best error correction protocols (*cascade*) corrects errors fairly well, in fact close to the Shannon limit, and the decrease in bit-rate of the protocol is around one half at modest error-rates (<5%). The decrease in bit-rate over the quantum channel, including the detector, is in the order of 10^{-2}. This indicates that work towards optimization of the latter part is desirable. The public communication part of quantum key distribution, investigated in this thesis, should be possible to implement to work at speeds high enough to not be the main bottleneck in the overall system performance. The most time-consuming part of the protocol seems to be privacy amplification, due to the complexity of the hash-functions. Further investigations towards faster processing of hash-function multiplication's, to speed

up the protocol performance are useful. However, time-delay due to computational processing is not directly comparable with the explicit loss in bit-rate of the protocol. It is enough for the protocol to be executed at least the speed of incoming rate of raw-bits, in order not to sink the effective bit-rate. To achieve this seems not to be a problem.

7.2 Future of QC?

Today as everyone is wiring up to the net the demand on security is increasing at a fast pace. Within trade, banking, etc. cryptography is more important than ever. When information is send from portable banking machines, or such, information has to be encrypted. It is important that no one who shouldn't can read the message. But it is even more important to be sure that it is the right person you are trusting on the other side of the communication line. There is a big need for secure communications. The main bottleneck of today's cryptography algorithms is their sensitivity to future improvements of computer speeds. If quantum computing ever comes true, or even if classical computers become enough faster than today, classical cryptography may have meet its match. Returning to Artur Ekert, whom we quoted in the introduction, he states, "*If a real need shows up, quantum cryptography can be implemented right away. If you really want to go for something that's perfectly secure, we know of no other option than quantum cryptography.*" So far so good, experimental results have show that indeed, we are not far from being able to implement systems with performance of commercial interest. There are several groups around the world working towards this goal right at this moment.

Bibliography

[1] J. Adàmek, "Foundation of Coding," *Wiley Interscience*, (1991), ISBN 0-471-62187-0.

[2] C. H. Bennett, "Quantum Cryptography using any two non-orthogonal states," *Physics Review Letter* 68, 3121 (1992).

[3] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, J. Smolin, "Experimental Quantum Cryptography," *J. of Cryptography v.*5, pp. 3-28 (1992).

[4] C. H. Bennett, G. Brassard, "Quantum Cryptography: Public-key distribution and coin tossing," *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, Bangalore, India. pp.175-179. (1984).

[5] C. H. Bennett, G. Brassard, C. Crépeau, U. M. Maurer, "Generalized Privacy Amplification", IEEE Trans. Info. Theory, 41, 1915 (1995).

[6] C. H. Bennett, G. Brassard, J-M. Robert, "How to reduce your enemy's information," *Advances in Cryptology - Proceedings of Crypto* '85, August 1985, Lecture Notes in Computer Science, Vol. 218, Springer-Verlag, Berlin, 1986, pp. 468-476.

[7] C. H. Bennett, G. Brassard, J-M. Robert, "Privacy Amplification by Public Discussion, " *SIAM Journal on Computing*, Vol. 17, No.2, April 1988, pp.210-229.

[8] M. Bourennane, F. Gibson, A. Hening, A. Karlsson, P. Jonsson, T. Tsegaye, D. Ljunggren, E. Sundberg. "Experiments on Long Wavelength (1550nm) "Plug and Play" Quantum Cryptography Systems," Submitted. Accepted for oral presentation, *CLEO/QELS* 99, Baltimore, (1999).

[9] G. Brassard, L. Salvail, "Secret-Key Reconciliation by Public Discussion," *Adventures in Cryptology*, *EUROCRYPT*93, Lecture Notes in *Computer Science*, vol. 765, Springer-Verlag. N.Y. pp.410-423. (1994).

[10] D. Bruß, N. Lütkenhaus, "Quantum Key Distribution: from Principles to Practicalities", `http://xxx.lanl.gov/archive/quant-ph`: *quant-ph*/9901061 (1998).

[11] C. Cashin, U. M. Maurer, "Linking Information Reconciliation and Privacy Amplification," *J. of Cryptography.* 10, 97 (1997).

[12] A. Ekert, "Quantum Cryptography based on Bell's theorem," *Physics Review Letter* 67, 661 (1991).

[13] Hamming, "Coding and Information Theory," *Prentice Hall*, *Englewood Cliffs*, *N.J.* 07632, ISBN 0-13-139072-4 (1986).

[14] R. J. Hughes, W. T. Buttler, P. G. Kwait, G. G. Luther, G. L. Morgan, J. E. Nordholt, C. G. Peterson, C. M. Simmons, "Secure Communications using Quantum Cryptography," Internal Report, *University of California*, *Los Alamos National Laboratory.* (1997).

[15] F. Gibson, "Experimental evaluation of Quantum Cryptography system for 1550nm," *Masters of Science Thesis.* Telia Research AB and Laboratory of Photonics and Microwave engineering, Dep. of Electronics, KTH. (1998).

[16] A. Karlsson, "Quantum Information Processing: Basic Ideas, Implementations, and Possible Impact on Communications," *ECEC/IOOC*'97 *Tutorial.* KTH (1998).

[17] N. Lütkenhaus, "Estimates for practical quantum cryptography", `http://xxx.lanl.gov/archive/quant-ph`: *quant-ph*/9806008 (1998).

[18] N. Lütkenhaus, "Security against eavesdropping in quantum cryptography," *Physics Review. A*, Vol. 54, pp. 97-111 (1996).

[19] C. Marand, P.D. Townsend, "Quantum Key Distribution over distances as long as 30 km, " *Optics Letters,* August 1995 / Vol. 20, No. 16, pp.1695-1697.

[20] A. Muller, T. Hertzog, B. Huttner, W. Tittel, H. Zbinden, N. Gisin, , "'Plug and Play' systems for quantum cryptography, " *Appl. Phys. Lett.* 70 (7), (1997).

[21] S. J. Phoenix and P. D. Townsend, "Quantum Cryptography: How to Beat the Code Breakers using Quantum Mechanics," *Contemporary Physics*, 1995, Vol. 36, No. 3, pp.165-195.

[22] J. G. Proakis, M. Salehi, "Communications systems Engineering," *Prentice Hall*, 1994, ISBN: 0-13-300625-5.

[23] G. Ribordy, J-D Gautier, N Gisin, O. Guinnard, H. Zbinden, "Automated "Plug & Play" Quantum Key Distribution," Submitted *Electronics Letters*, (1998).

[24] J-F. Riendeau, "Simulation de Protocoles de Cryptographie Quantique," *Master of Science Thesis*, *Universite de Monreal*, Départment d'Informatique et de Recherche Opérationelle Faculté des Arts et des Sciences. (1995).

[25] B. Slutsky, R. Rao, P-C Sun, L. Tancevski, S. Fainman, "Defense Frontier Analysis of Quantum Cryptographic Systems," UCSD (1992).

[26] B. Slutsky, "Key Distillation in Quantum Cryptography", Ph.D. Dissertation UCSD California (1998).

[27] B. Slutsky, P-C. Sun, Y. Mazurenko, R. Rao, Y. Fainman, "Effect of channel imperfection on the secrecy capacity of a quantum cryptographic system", *J. of Modern Optics*. 44, 953 (1997).

[28] B. Slutsky, R. Rao, P-C. Sun, Y. Fainman, "Security of quantum cryptography against individual attacks," *Physics Review A*, Volume 57, NR 4 (1998).

[29] B. Schneier, , "Applied Cryptography, Protocols, Algorithms, and Source Code in C,"John Wiley & Sons, ISBN 0-471-12845-7 (1996).

[30] D. Stinson, "Cryptography: Theory and Practice," CRC Press ISBN 0-8493-8521-0. (1995)

[31] L. Tancevski, B. Slutsky, R. Rao, S. Fainman, "Evaluation of the Cost of Error Correction in Quantum Cryptographic Transmission," *SPIE* Vol. 3228. (1997).

[32] P. D. Townsend, "Quantum Cryptography on Optical Fiber Networks," *Optical Fiber Technology*, Vol 4, 345-370 (1998).

[33] H. Zbinden, H. Bechmann-Pasquinucci, N. Gisin, G. Ribordy, "Quantum cryptography," *Appl. Physics B*, (1998).

[34] H. Zbinden, N. Gisin, B. Huttner, A. Muller, W. Tittel, "Practical Aspects of Quantum Cryptographic Key Distribution," submitted to *J. of Cryptography*. (1998).

[35] C. P. Williams, S. H. Clearwater, "Explorations in Quantum Computing," *Springer Verlag*, especially pp. 113-142, 163-178, (1997) ISBN 0-387-94768-X

[36] *Physics world* (March 1998), "SPECIAL ISSUE: Quantum Information," including: A. Zeilinger, "Fundamentals of quantum information,"
W. Tittel, G. Ribordy, N. Gisin, "Quantum cryptography,"
D. Deutsch, A. Ekert, "Quantum Computation,"
D. DiVincenzo B. Terhal, "Decoherence: the obstacle to quantum computation,"

Appendix A

Notations

Symbols:

ν : Frequency [Hz]
L : Fiber length
μ : Photon Intensity, NR of photons per pulse (gives v)
η : Quantum efficiency
α : Fiber attenuation constant
R_{dark}: Photon detector dark count rate
δT: Photon detector gate width

X : Alice's random variable for A
Y : Bob's random variable for B
Z : Eve's random variable for E

A : Alice's send string of bits
B : Bob's received string of bits
E : Eve's eavesdropped string of bits

V_r : Alice's and Bob's shared r.v. as raw-key, corresponds to N_r.
V_s : Alice's and Bob's shared r.v. after sifting, corresponds to N_s.
V_e : Alice's and Bob's shared r.v. after reconciliation, corresponds to N_e.
V_f : Alice's and Bob's shared r.v. after privacy ampl., corresponds to N_f.

N_r : Raw-key length, (Nr of bits)
N_s : Sifted-key length, (Nr of bits)
N_e : Reconciled key length (after error correction), (Nr of bits)
N_f : Final key length (after privacy amplification), (Nr of bits)

I_{AB}: Alice's and Bob's shared Information
I_E : Eve's estimated Information
I_{XY}: Alice's and Bob's shared *normalized* Information
I_Z : Eve's estimated *normalized* Information

e : error rate ($QBER$)
e_T : Nr of errors corrected

u : Eavesdropper information from Intercept/Resend attack on PQC
v : Eavesdropper information from Beamsplitting attack on PQC
$t = u+v$: Eavesdropper information from any attack on PQC
q : Eavesdropper information due to Error Correction leakage via PC
g : Extra safety margin, security parameter
s : Privacy amplification compression size

I_E^{tol} : Eve's maximum tolerated information on final key
p_{unafe}: probability that I_E^{tol} is exceeded
p_{safe}=1-p_{unsafe}: probability that I_E^{tol} is not exceeded

B : Bit Rate sent by Alice [bits/sec]
R : Raw Bit Rate at Bob [bits/sec]
R_{eff} : Effective Bit Rate (key creation rate) [bits/sec]

r_f : Fraction of Bits lost in fiber
r_s : Fraction of Bits lost due to Sifting
r_{ec} : Fraction of Bits lost due to Error Correction
r_{pa} : Fraction of Bits lost due to Privacy Amplification

Abbreviations:

PQC: Private Quantum Channel
PC: Public Channel

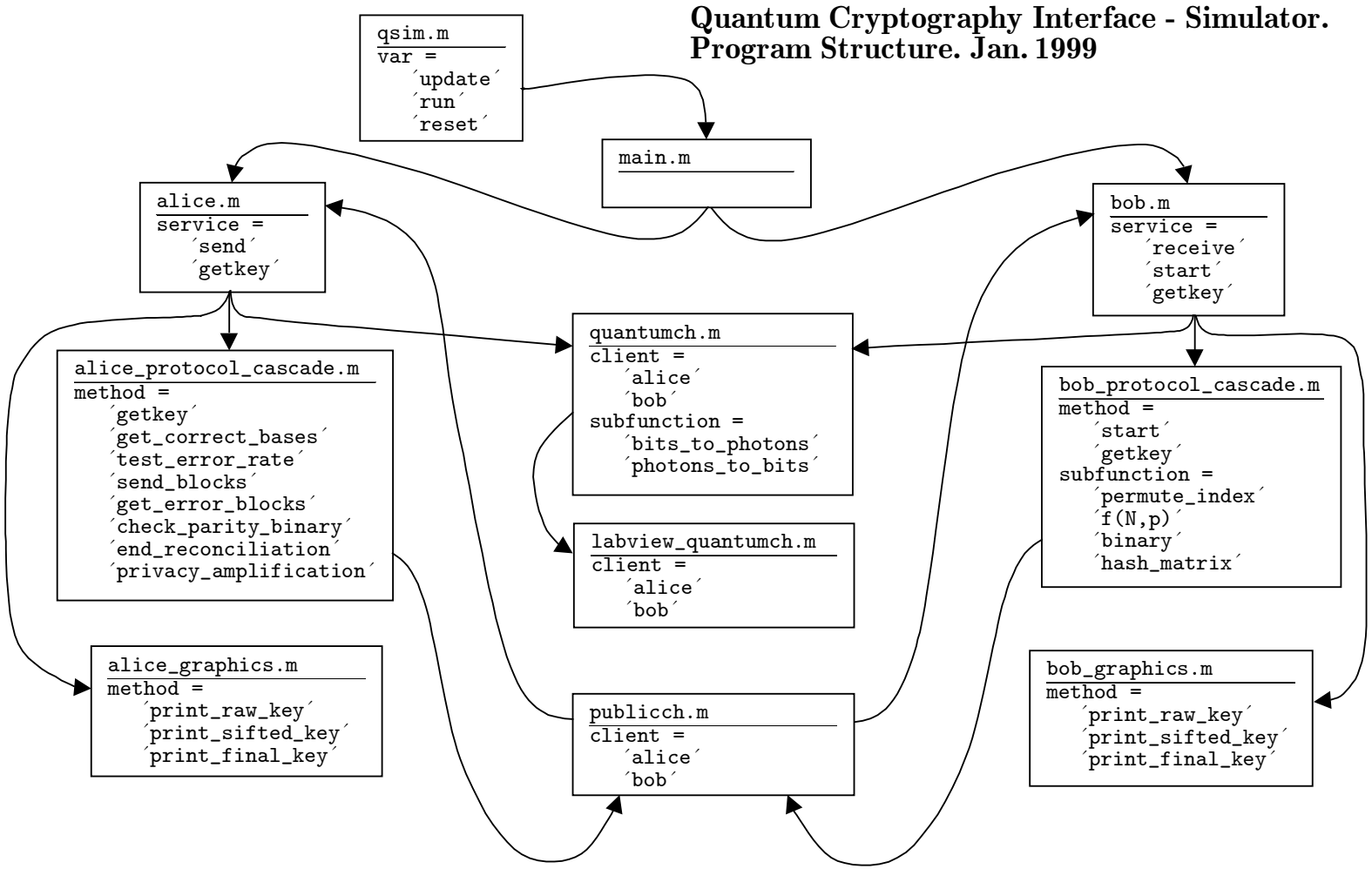
$QBER$: Quantum Bit Error Rate (for raw-key or sifted-key)
$EVEQBER$: Eve's injected Quantum Bit Error Probability on PQC
$TESTQBER$: Estimated overall Quantum Bit Error Rate after Sifting (gives t)
$ESTIMQBER$: Initial Test for Estimation of Quantum Bit Error Rate on PQC

BER: Bit Error Rate

Appendix B

Program Structure

The diagram shows the flow of function calls in the program. Functions are controlled by other functions by topics such as *client*, *service*, and *method*. Subfunctions are internal functions only visible within that function. Only functions of the *cascade* protocol implementation are shown.


Quantum Cryptography Interface - Simulator.
Program Structure. Jan. 1999
qsim.m
var =
´update´
´run´
´reset´
main.m
alice.m
service =
´send´
´getkey´
bob.m
service =
´receive´
´start´
´getkey´
quantumch.m
client =
´alice´
´bob´
subfunction =
´bits_to_photons´
´photons_to_bits´
alice_protocol_cascade.m
method =
´getkey´
´get_correct_bases´
´test_error_rate´
´send_blocks´
´get_error_blocks´
´check_parity_binary´
´end_reconciliation´
´privacy_amplification´
labview_quantumch.m
client =
´alice´
´bob´
bob_protocol_cascade.m
method =
´start´
´getkey´
subfunction =
´permute_index´
´f(N,p)´
´binary´
´hash_matrix´
alice_graphics.m
method =
´print_raw_key´
´print_sifted_key´
´print_final_key´
publicch.m
client =
´alice´
´bob´
bob_graphics.m
method =
´print_raw_key´
´print_sifted_key´
´print_final_key´