

Improved Secret Key Rate in  
Quantum Key Distribution  
using highly irregular  
Low-Density Parity-Check codes

Magne Gudmundsen

May 7, 2010



# Abstract

*Quantum key distribution (QKD)* has proven to be the most mature subfield of quantum information theory, elegantly spanning the gap between theory and practice. It is a simple, and yet sophisticated idea, in which the fundamental principles of modern quantum mechanics are employed to distribute a provably secure cryptographic key between two parties. With many variations of this scheme realized, it has become a question of enhancing each of the underlying structures in order to achieve optimal performance. One of these structures is the key reconciliation protocol, which has the objective of making sure the two communicating parties indeed have identical keys at the end of the protocol, by publicly exchanging an amount of extra information on the key. Since an antagonist can exploit this redundant information, the final key needs to be shortened with a corresponding amount.

The conventional approach to this problem has been for the two parties to engage in a dialogue over a secure channel, to locate and correct the faulty bits in their key. While such protocols are easy to implement and deliver stable performance, it is a time-consuming task with suboptimal capabilities in regards to the superfluous information made publicly available. When the lost *low-density parity-check (LDPC) codes* of Gallager were recently "rediscovered", it caused a boom in related research, generating reports of redundancy extremely close to the theoretical lower limit. Even though key reconciliation protocols based on these codes have been tried and tested for QKD, they have not been able to reproduce the performance of classical implementations and have thus enjoyed limited success.

The goal of this text is to demonstrate how, when suitably implemented, the true potential of LDPC codes can also be harnessed for QKD. After an extensive presentation of the theoretical background, an explicit key reconciliation protocol is presented. Simulations on a practical implementation are made, showing very good performance and outperforming some of the alternative protocols. The best code presented in this text operates at only 9.0% above the Shannon limit, while the interactive CASCADE protocol is reported to perform at  $\approx 21\%$  above, for the same set of parameters.

Additionally, the appreciable characteristics of the codes are utilized in a novel detection scheme, which promises tighter security at a lower cost on expended bits used for detecting an eavesdropper. This makes LDPC codes a highly competitive alternative to the conventional protocols.



## Acknowledgments

This report is the product of FY3900, Master thesis in Physics, as a partial fulfillment of the requirements to achieve a Masters degree at the Institute of Physics, NTNU. I would like to take the opportunity to thank my supervisor prof. Kåre Olaussen at the Institute of Physics, who gave me the opportunity to write about such an interesting topic.

The work on this paper was conducted in collaboration with the Quantum Hacking Group at NTNU, lead by prof. Johannes Skaar. As my external supervisor, he has been invaluable to my work with constant encouragement and advice, for which I am very grateful. Along with him I would like to thank my co-supervisor Øystein Marøy, as well as the rest of the quantum hackers for providing an inspiring research environment.

Since much of this work was performed in Bergen, I am very grateful to have been able to pick the brains of the good people working at Selmersenteret at the University in Bergen. Without the help of Joakim Grahl Knudsen and Eirik Rosnes my codes would forever diverge.

I would also like to show my appreciation to Hermund Torkildsen at the Department of Mathematics at NTNU, for enlightening discussions. And where would I be without my programming-guru, Stian Dahl, who is never too busy to answer the stupid questions? I have now learned to google before I ask.

Finally, I must thank my parents for showing an infinite amount of patience and support, and of course my dear Liv for being a constant source of inspiration and reminding me that there are more important things than quantum hacking.



# Variables

Table 1: Variables

Notation	Description	Section
$H(X)$	Shannon Entropy	2.1.1
$h(p)$	Binary entropy function	2.1.1
$r$	Information Rate	2.1.2
$n$	Block-size, length	2.1.2
$m$	Amount of redundant bits	2.1.2
$p$	Transition probability for a BSC	2.1.3
$p^*$	Estimated QBER	3.1.2
$p_{SL}$	The Shannon limit on $p$ , for rate $r$ .	4.5
$p_{DEL}$	Limit on $p$ according to density evolution	4.5
$p_{OL}$	The observed limit of decodability	5.3
$C(X)/C(p)$	Shannon Limit	2.1.3
$\hat{\mathbf{m}}$	Message vector	2.1.4
$\hat{\mathbf{k}}$	Key vector	2.1.4
$\hat{\mathbf{w}}$	Cipher vector	2.1.4
$\hat{\mathbf{x}}$	Original key vector	3.1.1
$\hat{\mathbf{y}}$	Measured key vector	3.1.1
$\hat{\mathbf{b}}/\tilde{\mathbf{b}}$	Basis vectors	3.1.1
$\hat{\mathbf{z}}$	Private key vector	3.1.5
$\hat{\mathbf{c}}$	Codeword vector	4.1
$\hat{\mathbf{e}}$	Error vector	4.1
$\hat{\mathbf{s}}$	Syndrome vector	4.1
$\rho$	Density Matrix	2.2.1
$\kappa(p)$	Efficiency of error-correction	3.1.4
$k$	Length of private key	3.1.5
$s$	Secret key rate	3.1.5
$\alpha$	Laser-dependent parameter	3.2
$C$	Linear code	4.1
$\mathbf{G}$	Generator Matrix	4.1
$\mathbf{H}$	Parity-check Matrix	4.1
$a$	Dimension of a code	4.1
$d_{\min}$	Minimum distance of a code	4.1
$l$	Cycle length, tree depth, iteration number	4.3.1
$g$	Girth	4.3.1
$d_c/d_v$	Regular Colum- / Row weight	4.3.2
$b_i/r_i$	Irregular degree distribution, node POV	4.3.2
$\lambda(x)/\rho(x)$	Irregular degree distribution, edge POV	4.3.2
$d_c^{\max}/d_v^{\max}$	Max check / variable node weight	4.3.2

# Abbreviations

Table 2: Acronyms

Abbreviation	Description	Section
<i>RSA</i>	Rivest-Shamir-Adleman, public cryptography	1.1
<i>BSC</i>	Binary Symmetric Channel	2.1.3
<i>BEC</i>	Binary Erasure Channel	4.5
<i>BIAWGNC</i>	Binary Input White Gaussian Noise Channel	4.5
<i>QKD</i>	Quantum Key Distribution	3
<i>BB84</i>	Bennet, Brassard, 84. QKD scheme	3.1
<i>B92</i>	Bennet, 92. QKD scheme	3.2
<i>P&amp;M</i>	Prepare & Measure	3
<i>EPR</i>	Einstein, Podolsky, Rosen	3
<i>QBER</i>	Quantum Bit-Error Rate	3.1.2
<i>PNS</i>	Photon Number Splitting - attack	3.2.1
<i>SPD</i>	Single Photon Detector	3.2.3
<i>APD</i>	Avalanche Photodiode	3.2.3
<i>DEM</i>	Detector Efficiency Mismatch	3.2.3
<i>ML</i>	Maximum Likelihood decoding	4.1
<i>LDPC</i>	Low-Density Parity-Check codes	4.2
<i>PEG</i>	Progressive Edge Growth, algorithm	4.3.3
<i>BP</i>	Belief Propagation	4.4.1
<i>LLR</i>	Log-Likelihood Ratio	4.4.2
<i>FEC</i>	Forward Error Correction	5.1
<i>OTP</i>	One-Time Pad	2.1.4

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Variables</b>	<b>v</b>
<b>Abbreviations</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Historical . . . . .	1
1.2 Motivation . . . . .	2
1.3 Terminology . . . . .	2
1.4 Progress of this text . . . . .	2
<b>2 Information Theory</b>	<b>5</b>
2.1 Classical Information Theory . . . . .	5
2.1.1 Entropy . . . . .	5
2.1.2 Rate of information . . . . .	6
2.1.3 Channel . . . . .	7
2.1.4 Cryptography . . . . .	7
2.1.5 Universal <sub>2</sub> functions . . . . .	9
2.2 Quantum Information Theory . . . . .	9
2.2.1 The qubit . . . . .	9
2.2.2 Measurements . . . . .	11
2.2.3 The no-cloning theorem . . . . .	12
<b>3 Quantum Key Distribution</b>	<b>13</b>
3.1 BB84-Protocol . . . . .	13
3.1.1 Photon transmission . . . . .	14
3.1.2 Estimation . . . . .	15
3.1.3 Detection . . . . .	16
3.1.4 Error-correction . . . . .	17
3.1.5 Privacy amplification . . . . .	19
3.2 Realization . . . . .	21
3.2.1 Source . . . . .	21
3.2.2 Channel . . . . .	23
3.2.3 Detectors . . . . .	23
3.3 Security . . . . .	25
3.3.1 Definition . . . . .	25
3.3.2 Proofs . . . . .	25

<b>4</b>	<b>Classical Coding Theory</b>	<b>29</b>
4.1	Linear Codes . . . . .	29
4.2	Low-Density Parity-Check Codes . . . . .	31
4.3	Construction . . . . .	32
4.3.1	Graphs . . . . .	32
4.3.2	Distributions . . . . .	34
4.3.3	Edge placement . . . . .	35
4.4	Decoding . . . . .	37
4.4.1	Belief propagation . . . . .	38
4.4.2	Log-likelihood ratio . . . . .	40
4.4.3	Complexity . . . . .	42
4.5	Density Evolution . . . . .	43
<b>5</b>	<b>Implementation</b>	<b>45</b>
5.1	Forward Error Correction . . . . .	45
5.2	Programming . . . . .	46
5.2.1	Decoding algorithm . . . . .	46
5.2.2	Codes . . . . .	48
5.2.3	Simulations . . . . .	48
5.3	Results . . . . .	48
<b>6</b>	<b>Discussion</b>	<b>53</b>
6.1	Kappa . . . . .	53
6.2	Speed . . . . .	55
6.3	Security . . . . .	56
6.4	Disadvantages . . . . .	59
6.5	Outlook . . . . .	60
<b>7</b>	<b>Concluding remarks</b>	<b>63</b>
<b>A</b>	<b>Codes</b>	<b>73</b>
<b>B</b>	<b>Source Code</b>	<b>77</b>

# 1 Introduction

## 1.1 Historical

Ever since the birth of quantum mechanics in the early twentieth century, its implications have engulfed just about every aspect of the way we interpret the world around us. From black holes to the inner workings of our brain, the growth of this new mindset seems to be endless. Hence, it was only a question of time before its fundamental ideas were put to use in the field of information theory and computer science.

The law of quantum superposition inspired Schumacher's introduction to the *qubit*[1], while the *no-cloning theorem* of Wootters and Zurek[2] proved another important milestone. Conceptual innovations such as these helped spawn entirely new fields of research called *quantum information theory* and *quantum computation*.

As soon as the concept of a quantum computer was born, scientists began investigating its applications. That led to groundbreaking contributions such as Shor's factorization algorithm[3] and Grover's search algorithm[4]. These algorithms are based on the practical use of quantum computers, which are still far from reality. One of the few practical implementations yet to emerge from this new field is quantum cryptography. Shor's factorization algorithm shook the field of classical cryptography to its core, postulating an efficient way of factorizing big numbers. The practical implementations of modern cryptography, like RSA[5] do not generally have unbreakable encryption, but rather makes the computation needed to break it extremely demanding. This is due to the fact that at the present time no efficient classical algorithm exists to factorize large numbers. Thus if quantum computers were to be built and Shor's algorithm implemented, this would mean that all of today's encryption would be threatened.

In 1984 Bennett and Brassard[6] proposed a brand new protocol for distributing a secure key based on quantum mechanical principle. The subfield of *quantum cryptography* was born. In stark contrast to the unproven security of classical encryption schemes, this new protocol promised *unconditional security* based solely on the subtle workings of quantum mechanics. This beautiful idea sparked intense research, making quantum cryptography to this day one of the only practically implementable subfields of quantum information theory.

Parallel to these advances, the field of coding theory was progressing steadily. However, in 1997 MacKay[7] rediscovered the "lost" concept of *low-density parity-check codes*. These had originally been proposed by Gallager in 1963[8], but had been discarded due to the insufficient computational power at the time, and thus largely forgotten. Their remarkable performance was undeniable, and led to intense research.

## 1.2 Motivation

The process of quantum key distribution is dependent on an *error-correction-protocol*, which reconciles an initially flawed key shared by the two communicating parties. The conventional CASCADE-protocol[9] has long been known to be suboptimal in many regards, but LDPC codes have not yet been shown to be sufficiently proficient in order to be widely implemented. Even though classical coding-theory reports performance of LDPC codes extremely close to the theoretical limit[10], the adaptations used for QKD have not yet delivered results that come close to these capabilities[11, 12]. Closing this gap between the true potential of LDPC codes and the observed performance would surely benefit the field of QKD, and will be the topic of this text.

## 1.3 Terminology

The contents of this text span different academic sectors and for this reason certain expressions and terms might prove ambiguous. Therefore it is useful to introduce the terminology and notation to be used throughout this text.

Contrary to traditional mathematical conventions, vectors are in this text represented in row-form,  $\hat{\mathbf{x}} = [x_1x_2x_3 \dots x_n]$ . When using mathematics to describe information, expressions such as *messages* and *codewords* are more intuitively understood with such a notation. Unless stated otherwise, the default alphabet for all vectors and matrices will be binary.

The *weight* of a vector  $\hat{\mathbf{x}}$  is defined as the number of non-zero entries, which in the binary case becomes the number of ones, and is denoted  $\text{wt}(\hat{\mathbf{x}})$ . In many applications it is useful to know how far two vectors are apart, hence a practical definition is the *distance*, also known as the *Hamming distance*, between two vectors  $\hat{\mathbf{x}}$  and  $\hat{\mathbf{y}}$ ,  $d(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ .

$$d(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \text{wt}(\hat{\mathbf{x}} \oplus \hat{\mathbf{y}}) \tag{1.1}$$

The summation,  $\oplus$ , denotes a bitwise modulo two summation.

The *density* of a matrix is defined as the fraction of non-zero entries divided by the total size of the matrix.

## 1.4 Progress of this text

In the sequel of this text I will begin by giving a brief introduction to the notions of information theory, both from a classical and quantum mechanical point of view in Section 2, and reveal some of the subtle differences that make quantum key distribution possible. This will be employed in Section 3, where I proceed to broadly explain the principles of quantum key distribution. Several aspects are

reviewed, including practical implementations and security issues, with the main focus on how an improved key reconciliation process can be beneficial in many ways. One of the main contributions of this text is the comprehensive study of the underlying theory of low-density parity-check codes in Section 4. With it, I show both why and how these codes can achieve the excellent performance they are known for.

In Section 5 I tie these two fields together and show how the LDPC codes can be modified to be used for quantum key distribution. This is done both from a theoretical point of view, and practically through the description of my implementations and simulation results. Based on this strong foundation, I argue in Section 6 how an improved key reconciliation protocol can be achieved when the true potential of these codes are fully harnessed. I present a working protocol, and a novel detection scheme.



## 2 Information Theory

### 2.1 Classical Information Theory

Given the two parallel but widely different approaches to information theory, it is important to clearly separate the two schools of thought: classical and quantum information theory.

#### 2.1.1 Entropy

In 1949, Claude Shannon[13] published his seminal paper on the mathematical approach to information theory. It has since been cited in excess of 7.000 times and is one of the principle pillars of modern information theory. Among many valuable contributions, he formulated a mathematical measure of uncertainty in receiving a variable, later dubbed the *Shannon Entropy*. For a variable  $X$  with  $n$  possible outcomes, each occurring with probability  $p_i$ , the Shannon entropy of  $X$  is defined in the following way:

$$H(X) = - \sum_{i=1}^n p_i \log_2(p_i). \quad (2.1)$$

This quantity can be interpreted in two seemingly different but ultimately complementary ways. For a random variable  $X$ , Nielsen and Chuang[14] describes the two approaches to  $H(X)$  as such:

*"The Shannon entropy of  $X$  quantifies how much information we gain, on average, when we learn the value of  $X$ . An alternative view is that the entropy of  $X$  measures the amount of uncertainty about  $X$  before we learn its value."*[14]

In the binary world, where we operate in this text, the Shannon entropy is reduced to only two terms and is referred to as the *binary entropy function*,  $h(p)$ , as illustrated in Fig. 2.1.

$$h(p) = -p \log_2(p) - (1 - p) \log_2(1 - p) \quad (2.2)$$

The variable  $p$  in this instance is the a priori probability that the bit is a 0, which makes  $1-p$  the complementary probability of a 1. The figure shows  $h(p)$  approaching 0 as  $p$  approaches 0 or 1, which is intuitive since the receiver gains no information by a bit which is either 0 or 1 with unit probability. Mathematically  $\log_2(0)$  is undefined so the convention  $0 \log_2(0) = 0$  is applied. For  $p = \frac{1}{2}$  both bit-values are equiprobable and its inherent information reaches its peak value as  $h(\frac{1}{2}) = 1$ .

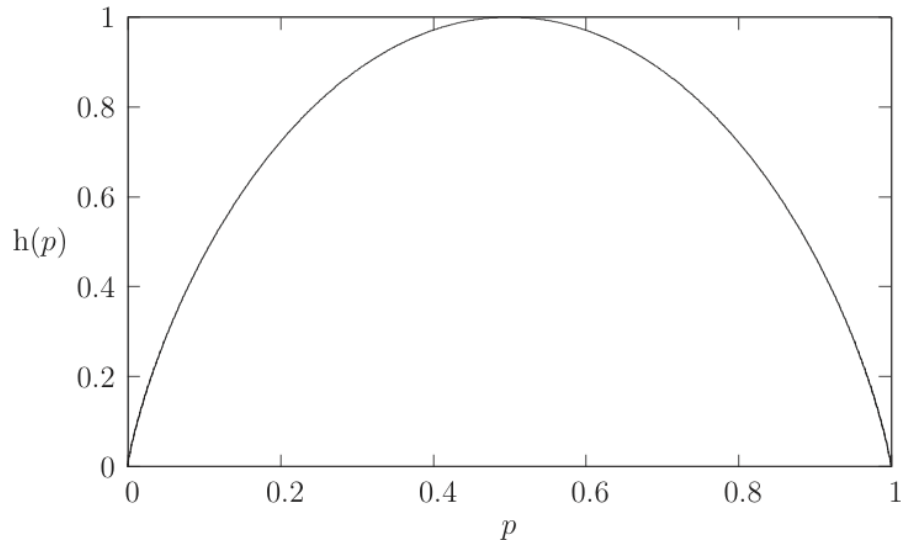


Figure 2.1: Binary Entropy function

### 2.1.2 Rate of information

To make sure that a message is correctly acquired by the receiver when transmitted over a noisy channel, the sender also has to provide some extra information about the message to compensate for the potential errors in the received message. The amount of *redundant bits*,  $m$ , is dependent on the amount of noise in the channel, the length of the message,  $n$ , and the *error-correction protocol* applied. For any protocol the ratio between actual *information carrying bits* to the total amount of bits sent is known as the *information rate*, or just *rate*,  $r$ .

$$r = \frac{n - m}{n} = 1 - \frac{m}{n} \quad (2.3)$$

In his *noisy channel coding theorem*, Shannon states that for every channel there exists an upper bound on the information rate called the *Shannon limit*,  $C(X)$ . This means that if  $r < C(X)$  an error-correction scheme exists that can reliably transmit information with an arbitrarily small failure probability in the limit  $n \rightarrow \infty$ . Conversely, any attempt at transmitting information with  $r > C(X)$  will have a failure probability bounded away from zero and positively dependent on  $r$ . Using the same alphabet as in Eq. (2.1), the Shannon limit is defined in the following way:

$$C(X) = 1 - H(X) = 1 + \sum_{i=1}^n p_i \log_2(p_i). \quad (2.4)$$

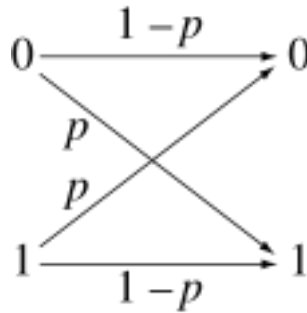


Figure 2.2: Binary Symmetric Channel

The field of *coding theory*, which will be the topic of section 4, is constantly searching for error-correction schemes that can approach this limit.

### 2.1.3 Channel

A key aspect in evaluating the transfer of information between two parties, is the channel over which it takes place. One of the simplest models for a channel is the *binary symmetric channel (BSC)*, which will be an important model in the later stages of this text. It takes binary input  $x_i$  and flips it with a certain probability,  $p$  and leaves it with probability  $1 - p$ , so that the output,  $y_i$ , follows the statistics  $P(y_i = 0|x_i = 1) = P(y_i = 1|x_i = 0) = p$  and  $P(y_i = 0|x_i = 0) = P(y_i = 1|x_i = 1) = 1 - p$ , as illustrated in Fig. 2.2.

Consequently, a BSC is fully characterized by the transition probability  $p$ , conventionally denoted  $\text{BSC}(p)$ . It is also assumed memoryless with a uniform distribution of errors.

In order to describe the information rate possible over a  $\text{BSC}(p)$  it is useful to associate the uncertainty associated with every received bit to the quantity  $h(p)$ . Since the Shannon entropy can be regarded as a measure of uncertainty, the corresponding Shannon limit for a  $\text{BSC}(p)$  becomes:

$$C(p) = 1 + p \log_2(p) + (1 - p) \log_2(1 - p). \quad (2.5)$$

### 2.1.4 Cryptography

The field of *cryptology* concerns the problem of secretly exchanging information between two or more parties and should not be confused with coding theory. The practice of encoding messages into codewords, as will be described in Section 4, does not in any way provide security against an antagonist.

<b>Alice</b>		<b>Bob</b>
$\hat{\mathbf{m}}$ 11010011		
$\oplus \hat{\mathbf{k}}$ 01101010		
$\hat{\mathbf{w}}$ 10111001	→	$\hat{\mathbf{w}}$ 10111001
		$\oplus \hat{\mathbf{k}}$ 01101010
		$\hat{\mathbf{m}}$ 11010011

Figure 2.3: One-time pad

In the following the focus will be on the case of two parties, conventionally called *Alice* and *Bob*, who want to communicate securely while constantly being challenged by the evil eavesdropping *Eve*. While keeping secrets has always been a part of human nature, the development of ever more sophisticated means of communication called for more formal definitions of secrecy and privacy. The pioneering contribution of Vernam in 1926[15] proved a milestone within this field. He proposed a protocol where Alice and Bob use a pre-shared secret *key*,  $\hat{\mathbf{k}}$ , to encrypt and decrypt a message,  $\hat{\mathbf{m}}$ . This protocol, named the *vernam cipher* after its inventor, is applicable with any alphabet but will in this text be used solely with the binary alphabet.

The key and message both need to be of equal length,  $n$ . The process of encryption is performed using a bit-by-bit *XOR* operation on  $\hat{\mathbf{k}}$  and  $\hat{\mathbf{m}}$ , resulting in a *cipher*  $\hat{\mathbf{w}}$ . Alice sends  $\hat{\mathbf{w}}$  to Bob, who in turn uses the same key to extract the original message in an equivalent operation, as illustrated by Fig. 2.3.

Eve has in this case full access to the encrypted cipher,  $\hat{\mathbf{w}}$ , so if the same key was to be used in several sessions she could theoretically deduce certain attributes of the key. One such instance could be if the same key was first used to encrypt the email-header, and later the content of the email. Since headers usually follow a set pattern, Eve could find the values of certain, if not all, bits of the key. This would render the key absolutely useless in subsequent encryption sessions, and leave the contents of the email unsecured. Hence, it is of paramount importance that the key is discarded after use, giving the protocol its more widely used name *the one-time pad (OTP)*.

In 1949, Shannon[16] proved that a key of equal length as the message provides *perfect security*, meaning that an adversary would be unable to break the encryption without placing any restrictions on her capabilities.

Practical applications of one-time pads however, remain nonviable because of the strict requirements of perfect randomness of the pads as well as the problem of

securely and secretly distributing them. Both of these obstacles are elegantly addressed by the advent of *quantum key distribution*, a field which will be thoroughly explored in this text.

### 2.1.5 Universal<sub>2</sub> functions

The label *universal<sub>2</sub>*[17] is normally given to a set of hash functions,  $\mathcal{F}$ , from  $\mathcal{X}$  to  $\mathcal{Y}$ , with the property that any function  $f \in \mathcal{F}$  should have a very low probability of taking two distinct elements of  $\mathcal{X}$  to the same element of  $\mathcal{Y}$ . The formal definition is as follows:

$$\text{Prob}(f(x_i) = f(x_j)) \leq \frac{1}{|\mathcal{Y}|}, \quad \forall \{x_i \neq x_j\} \in \mathcal{X}, \quad \forall f \in \mathcal{F}, \quad (2.6)$$

where  $f$  is chosen with uniform probability from  $\mathcal{F}$ .

Due to this property, universal<sub>2</sub> hash functions are often used to test for equality between two variables and makes them ideal for use in authentication schemes[18].

A trivial example would be the set of all functions from  $\mathcal{X}$  to  $\mathcal{Y}$ . Because two parties usually need to agree upon a single function, the amount of information needed to be exchanged in order to unambiguously describe a single function from such a big set is unfeasible. Hence, several more compact sets have been proposed[17].

## 2.2 Quantum Information Theory

This text will assume basic knowledge of quantum mechanics. However, due to the distinctive approach adopted by the field of quantum information theory, a brief introduction is beneficial. The following introduction follows that of Nielsen and Chuang[14], which should be consulted for a more comprehensive study.

### 2.2.1 The qubit

Contrary to the classical notion of a bit, taking either the discrete values 0 or 1, Schumacher's[1] quantum mechanical *qubit* can exist in a continuous space between zero and one due to the principle of quantum mechanical *superposition*. Using the *bracket notation* of Dirac, two separate quantum states of a system,  $|\phi\rangle$  and  $|\psi\rangle$ , can be arbitrarily attributed the logical zero and one states,  $|0\rangle$  and  $|1\rangle$ . Thus, in the same way that the state of a system can be in a quantum mechanical superposition,  $|\Psi\rangle = \alpha|\phi\rangle + \beta|\psi\rangle$ , a qubit can "be anywhere between" a zero and one,  $|\hat{x}_i\rangle = \alpha|0\rangle + \beta|1\rangle$ , where the factors  $\alpha$  and  $\beta$  are probability amplitudes, satisfying the constraint  $|\alpha|^2 + |\beta|^2 = 1$ .

Since the superposition principle is a general feature of quantum mechanics, any quantum system that has two separate states can constitute a qubit. Which quantum systems that are practical for use as qubits however, is largely dependent on the purpose of the system. It is desirable that a qubit can easily be manipulated in order to do computation, however it is also important that the qubit remains in its state and retains its information when not acted upon. From a quantum mechanical perspective this means finding a system that easily interacts with its environment in specific situations that are induced by an experimentalist, but rarely interacts with its environment otherwise. Additionally, it is essential to be able to prepare a qubit in a specific state and measure it reliably. Finding such systems is an extremely difficult experimental problem that is out of the scope of this text.

In the following we will regard the four general states, with their respective inner-products:

$$\left. \begin{array}{l} |0\rangle \\ |1\rangle \end{array} \right\} \langle i|j\rangle = \delta_{i,j} \text{ for } i, j \in \{0, 1\}$$
$$\left. \begin{array}{l} |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{array} \right\} \langle i|j\rangle = \delta_{i,j} \text{ for } i, j \in \{+, -\},$$

Notice how these two sets of states,  $\{|0\rangle, |1\rangle\}$  and  $\{|+\rangle, |-\rangle\}$ , are separately orthonormal, which makes them ideal as bases for the two-dimensional state space spanned by a qubit, respectively denoted the Z- and X- basis. However, since  $\langle 0|\pm\rangle = \frac{1}{\sqrt{2}}$  and  $\langle 1|\pm\rangle = \frac{\pm 1}{\sqrt{2}}$ , Heisenberg's uncertainty principle denies accurate knowledge of both observables simultaneously, which will be investigated further in the following section.

A *pure state* of a system is one that can be expressed with a ket,  $|\Psi\rangle$ . Nevertheless, in quantum information theory it is often required to express the state of a system in a more general way using a *density matrix*,  $\rho$ . If a system can occupy several different pure states,  $\{|\Psi\rangle_i\}$ , with corresponding probabilities,  $\{p_i\}$ , the density of the matrix is defined as such:

$$\rho = \sum_i p_i |\Psi\rangle_i \langle \Psi|_i. \quad (2.7)$$

One of the main advantages of working with the density matrix is because it is easy to distill individual subsystems from a composite system. If  $\rho_{AB}$  denotes the state of the joint system of A and B, then the state of A can be found by *tracing out* the system of B.

$$\rho_A = \text{Tr}_B(\rho_{AB}). \quad (2.8)$$

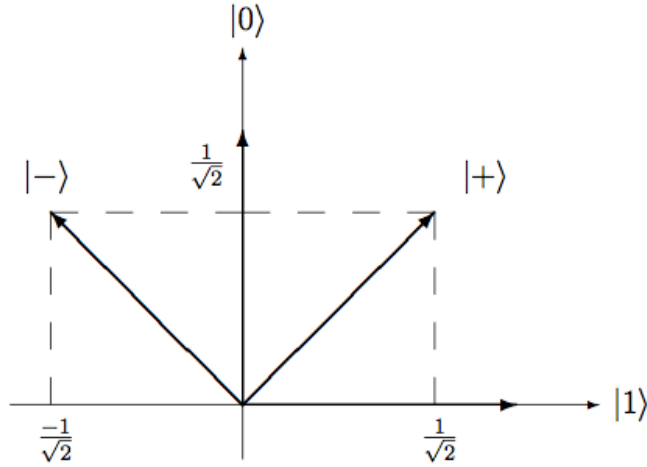


Figure 2.4: Two-dimensional state-space of a qubit

The operator  $\text{Tr}_B$  in this case is the partial trace over the subsystem of B.

### 2.2.2 Measurements

Conducting measurements in this setting is commonly performed using a set of orthogonal *projection measurements*,  $\{P_t\}$ , which project onto the eigenspaces of the system,  $|\Psi\rangle$ , with eigenvalues  $t$ . The state immediately after such a measurement, given that the measurement outcome is  $t$ , is given by  $P_t|\Psi\rangle/\sqrt{p(t)}$ , where  $p(t) = \langle\Psi|P_t|\Psi\rangle$  denotes the probability of measurement outcome  $t$ .

The two sets of states considered above can be used as bases for the two-dimensional system of a qubit, and can consequently be used in a projective measurement. This gives the two projections measurements:  $\{P_0 = |0\rangle\langle 0|, P_1 = |1\rangle\langle 1|\}$  and  $\{P_+ = |+\rangle\langle +|, P_- = |-\rangle\langle -|\}$

A general state of a qubit,  $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ , will give the following measurement statistics when measured in the Z-basis:

$$p(0) = \langle\Psi|P_0|\Psi\rangle = (\langle 0|\alpha^* + \langle 1|\beta^*)|0\rangle\langle 0|(\alpha|0\rangle + \beta|1\rangle) = |\alpha|^2 \quad (2.9)$$

$$p(1) = \langle\Psi|P_1|\Psi\rangle = (\langle 0|\alpha^* + \langle 1|\beta^*)|1\rangle\langle 1|(\alpha|0\rangle + \beta|1\rangle) = |\beta|^2 \quad (2.10)$$

which gives the expected relation  $p(0) + p(1) = |\alpha|^2 + |\beta|^2 = 1$ . Such a measurement leaves the state of the system in the state  $|0\rangle$  or  $|1\rangle$  corresponding to the measurement outcome.

It is then easily deduced that a qubit in the state  $|0\rangle$  or  $|1\rangle$  will respectively give the measurement statistics  $p(0) = 1, p(1) = 0$  or  $p(0) = 0, p(1) = 1$ , when measured in the Z-basis, and  $p(0) = p(1) = \frac{1}{2}$  when measured in the X-basis. For the states  $|+\rangle$  and  $|-\rangle$ , measurements in the same bases as above will give the converse statistics.

As mentioned above, any measurement will leave the system in the state corresponding to the measurement outcome. Hence, an experimentalist measuring the state of a system in a basis non-orthogonal to the basis in which the state was prepared, would necessarily perturb the state. E.g. a system in the state  $|+\rangle$  or  $|-\rangle$  that is measured in the Z-basis will end up in the state  $|0\rangle$  or  $|1\rangle$ , with probability  $\frac{1}{2}$  for each.

### 2.2.3 The no-cloning theorem

A fundamental feature of quantum mechanics, which is of paramount importance to the field of quantum key distribution, is the so-called *no-cloning theorem*. First proved by Wootters and Zurek[2], it states that it is impossible to clone an unknown quantum state.

To prove this, let us assume Bob starts off with a random state  $|\chi\rangle_B$  and wants to copy some state of Alice, leaving it intact, through a unitary evolution acting on their combined state. Assuming further that his process works for two random states of Alice,  $|\psi\rangle_A$  and  $|\phi\rangle_A$ , the following deduction holds.

$$U(|\chi\rangle_B \otimes |\psi\rangle_A) \longrightarrow |\psi\rangle_B \otimes |\psi\rangle_A \quad (2.11)$$

$$U(|\chi\rangle_B \otimes |\phi\rangle_A) \longrightarrow |\phi\rangle_B \otimes |\phi\rangle_A \quad (2.12)$$

By simply taking the inner product of both sides we get the following expression:

$$\langle\psi|_A \langle\chi|_B \underbrace{U^\dagger U}_I |\chi\rangle_B |\phi\rangle_A = \langle\psi|_A \langle\psi|_B |\phi\rangle_B |\phi\rangle_A \quad (2.13)$$

$$\langle\psi|\phi\rangle_A \underbrace{\langle\chi|\chi\rangle_B}_1 = \langle\psi|\phi\rangle_A \langle\psi|\phi\rangle_B \quad (2.14)$$

$$\langle\psi|\phi\rangle = (\langle\psi|\phi\rangle)^2. \quad (2.15)$$

This equation has only two possible solutions:

$$\left. \begin{array}{l} \bullet \langle\psi|\phi\rangle = \pm 1 \Rightarrow |\phi\rangle = \pm|\psi\rangle \\ \bullet \langle\psi|\phi\rangle = 0 \Rightarrow |\phi\rangle \perp |\psi\rangle \end{array} \right\} \langle\psi|\phi\rangle = \pm\delta_{\psi,\phi}$$

Thus, any such attempt to copy an unknown qubit is futile if Alice utilizes non-orthogonal quantum states.

### 3 Quantum Key Distribution

The revolutionizing idea proposed by Bennett and Brassard in 1984[6] was to incorporate fundamental laws of quantum mechanics to the field of cryptography. By exploiting the random nature of quantum mechanics as well as the no-cloning theorem, they introduced a novel way of secretly distributing a completely random key between the antagonists Alice and Bob, which can subsequently be used in an OTP-scheme. It has subsequently been known as the *BB84-protocol*, and even though several sophisticated protocols have been proposed it has come to be the prevailing choice for both theoretical and experimental work due to its simplicity.

The BB84-protocol is a so-called *prepare & measure (P&M)*-protocol, where the photons are prepared by one party, sent across and measured by another party. The 4-state-setup used in the BB84 has been both extended to a *6-state protocol*[19, 20] and reduced to a *2-state* one[21]. A conceptually different approach is the *entanglement-based* protocol of Ekert[22]. It was developed independently, and has since been proven to be equivalent to the BB84 from a security perspective[23]. Ekert proposed to prepare a series of *EPR-pairs*[24], of two entangled photons at a common trusted source and then distribute them to each of the two parties. While it is a more challenging protocol to realize, it has proven an extremely valuable contribution from a theoretical point of view.

Contrary to classical cryptography, which uses notions of information theory and computational complexity to argue that their schemes are safe, the security of QKD is based on the laws of physics. Hence, assuming that our present laws of quantum physics holds, cryptography employing the BB84-protocol can in theory be proven *unconditionally secure*, as will be shown in Section 3.3.

#### 3.1 BB84-Protocol

Alice and Bob are connected with two separate communication channels, a classical channel and a quantum channel. By assumption, the classical channel is lossless and authenticated, in which case evil eavesdropping Eve can listen to all the information being transmitted but not intercept or alter it. This scenario is illustrated in Fig. 3.1.

If the channel does not have any inherent authentication schemes, it is easy to utilize the universal<sub>2</sub> functions from Section 2.1.5 to authenticate the messages, consuming  $\approx \log_2(n)$  secret bits for an  $n$ -bit message[18]. Consequently, Alice and Bob need to share a certain amount of secret key prior to any key distribution protocol. Thus, using the name *quantum key expansion* is perhaps more fitting.

The quantum channel on the other hand, is open to any attack Eve wishes, limited only by the laws of physics.

In the original paper Bennet and Brassard[6] proposed using polarized photons

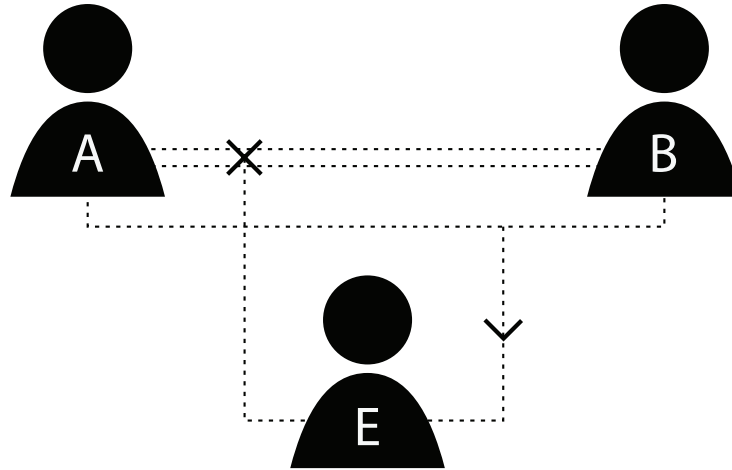


Figure 3.1: Alice (A) is connected to Bob (B) through two channels: the quantum channel (double line) and the public channel (single line). The eavesdropper Eve (E) has full access to the quantum channel but can only listen to the public channel

as qubits with four different polarization angles,  $[0^\circ, 90^\circ, 45^\circ, 135^\circ]$ . Even though several other realizations of qubits are possible, quantum key distribution consistently use photons[25]. This is due to photons not interacting easily with matter and thus being able to keep their quantum information over longer distances.

### 3.1.1 Photon transmission

The protocol starts with Alice generating two random bit arrays,  $\hat{\mathbf{x}}$  and  $\hat{\mathbf{b}}$ , of length  $2(n + \delta)$ , which is used to generate qubits. The first array corresponds to the bit-value of the qubit and the other to the basis in which it is encoded. With  $\hat{\mathbf{b}}_i$  deciding the basis in which qubit  $i$  is to be decoded, the Z-basis or the X-basis, the resulting qubit is subject to the following description:

$$\hat{\mathbf{b}}_i = 0 \begin{cases} \hat{\mathbf{x}}_i = 0 \implies |0\rangle \\ \hat{\mathbf{x}}_i = 1 \implies |1\rangle \end{cases}$$
$$\hat{\mathbf{b}}_i = 1 \begin{cases} \hat{\mathbf{x}}_i = 0 \implies |+\rangle \\ \hat{\mathbf{x}}_i = 1 \implies |-\rangle. \end{cases}$$

Now Bob on his side has to independently choose which basis to measure in, resulting in the bit-strings  $\tilde{\mathbf{b}}$  and  $\tilde{\mathbf{y}}$ . They correspond to Bob's choice of basis and measurement outcome respectively. After Bob is done measuring the qubits,

he publicly announces this fact to Alice. This and all subsequent transmission of information between the two parties is conducted over the public channel, and constitutes what is known as the classical *post-processing*. All the following steps are also symmetrical with regards to Alice and Bob. It is purely by convention that Alice's key is arbitrarily designated as the correct key, which Bob has to emulate.

Only after Bob declares that he has measured the preset number of photons does Alice announce her choice of basis by sending  $\hat{\mathbf{b}}$  to Bob. Obviously Bob will have guessed the wrong basis about half the time, and as previously described in Section 2.2.2 such measurements give no information since the outcome is completely random. These bits are thus uninformative and of no use to Alice and Bob, who proceed to remove them. This is handled through a process called *sifting*, where Bob compares  $\hat{\mathbf{b}}$  to  $\tilde{\mathbf{b}}$  discarding all the measurements where the bases do not coincide,  $\hat{\mathbf{b}}_i \neq \tilde{\mathbf{b}}_i$ . The resulting array of bits is called the *sifted-key*, with length  $\approx n + \delta$ .

### 3.1.2 Estimation

The quantum channel, regardless of physical implementation, is inherently faulty due to a spectrum of different errors to be discussed in Section 3.2. All errors that occur through the quantum channel, both caused by Eve and suboptimal hardware, are conventionally combined in the parameter *quantum bit-error rate (QBER)*. Because Eve in principle has total control over the quantum channel, all QBER needs to be attributed to her. An assumption that some of the observed QBER is due to the channel opens up a possible exploit for Eve. She could in theory replace the quantum channel with a superior one with lower error-rate and could thus incur a correspondingly increased amount of errors without being detected. However unlikely an attack may seem, all theoretical loopholes must be countered for a QKD-protocol to be provably secure.

Since the quantum channel is assumed to be a binary symmetrical channel as described in Section 2.1.3, the QBER corresponds to the transition probability of the channel,  $p$ .

In the ideal case with no eavesdropper and perfect equipment, Alice and Bob would at this point be in possession of their own version of a perfect key, both secret and identical. However, since this is not the case in the real world, Alice and Bob has to take some steps in order to reconcile their keys, which have been perturbed by what we have to assume is Eve manipulating the quantum line.

The first step in this process is to estimate the QBER on the shared key by randomly sampling a fraction,  $\delta$ , and publicly announcing the measurement results of these qubits. A convenient preparation called *uniformization* proposed by Bennett, Brassard and Robert[26], permutes the key-bits in a random, but public way. That way any attempt from Eve to introduce a burst of errors will be

thwarted, since the errors will be uniformly distributed across the key. This is also essential in order to approximate the quantum channel to the theoretical model of a binary symmetric channel, which is inherently memoryless.

Now Alice and Bob can simply sample the first  $\delta$  bits and find an estimate for  $p$ ,  $p^*$ .

$$p^* = \frac{\sum_{i=1}^{\delta} \hat{\mathbf{x}}_i \oplus \hat{\mathbf{y}}_i}{\delta} \quad (3.1)$$

Since the true values of the sampled bits were publicly announced to Eve they obviously have to be discarded, leaving Alice and Bob with reduced keys of size,  $|\hat{\mathbf{x}}| = |\hat{\mathbf{y}}| \approx n$ . It is therefore in the best interest of Alice and Bob to minimize the fraction  $\delta$ .

### 3.1.3 Detection

The true strength of quantum key distribution is revealed at this point. In classical communication there is no way for Alice and Bob to detect an eavesdropper on their line from their mutual information only because there is no law denying Eve the ability to just copy the bits being transmitted[27]. However, in quantum cryptography an eavesdropper will invariably perturb the state in a detectable way when Alice employs randomly chosen conjugate bases, as discussed in Section 2.2.2.

To exemplify this fundamental difference it is useful to consider the most basic attack Eve can perform, called the *intercept-resend* attack. As the name implies, Eve intercepts all, or some of, the qubits from Alice and measures them in a basis of her choice. She then proceeds to retransmit qubits to Bob corresponding to her measurement outcome. In half of the cases where she chose the correct basis she gets full information on the qubit and leaves it intact, but in the other half her measurement outcome is completely random leaving the qubit in a state non-compliant with the original basis. If Bob now measures this qubit in the original basis corresponding to Alice's, he has a 50% chance of getting an incorrect bit-value. Consequently, Eve can obtain full information on about 50% of the bits, but at the penalty of introducing about 25% errors. The estimation procedure described above will detect the discrepancies,  $p^* = 0.25$ , and Alice and Bob is faced with the decision if they want to try to proceed and try to distill a secret key or abort the protocol. This particular attack leaves Eve with more mutual information with  $\hat{\mathbf{x}}$  than Bob, forcing Alice and Bob to abort the protocol[25].

This ability to detect Eve's presence is the most important feature of quantum key distribution and allows the protocol to be provably safe. In principle there is nothing preventing Eve from stopping all the photons, obstructing the protocol altogether. However, the following post-processing steps, along with an upper

limit on  $p^*$ , make sure that Eve cannot gain any information on  $\hat{\mathbf{x}}$  without Alice and Bob knowing.

The general problem of deciding a tolerable  $p^*$  for which the pair can proceed is dependent on the possible attacks Eve can perform, which will be discussed further in Section 3.3. If  $p^*$  is below this limit, the pair can proceed to distill a secret identical key from the sifted key, through the processes of *error-correction* and *privacy amplification*.

### 3.1.4 Error-correction

Given an observed QBER below the tolerable limit, Alice and Bob can proceed to correct their remaining faulty key through the process of *key reconciliation*. The information Bob has on the correct key can now be expressed using the Shannon entropy:

$$H(\hat{\mathbf{x}} | \hat{\mathbf{y}}) = n \cdot (1 - h(p)) \quad (3.2)$$

Hence, Alice has to send at least the amount  $nh(p)$  of additional information to Bob for him to be able to correct the message according to Shannon in Section 2.1.2. Unfortunately, Shannon's proofs are non-constructive and practical error-correction protocols need more information than this, implying that  $h(p) \rightarrow \kappa(p)h(p)$ , where  $\kappa(p)$  is a factor  $> 1$  describing the efficiency of the protocol.

The process of transmitting this redundant parity information can be done in several different ways. The different available protocols are usually classified as either *interactive* or *non-interactive*, also called *two-way* and *one-way* protocols. These labels correspond to the pattern of communication between the two parties across the public channel. While the interactive protocols rely on heavy traffic of parity information back and forth in order to locate and correct errors, one-way reconciliation makes it possible for Alice to assemble and transmit only one set of parity information on the correct key.

Brassard and Salvail[9] with their CASCADE-protocol first introduced the idea of an interactive reconciliation protocol in 1993, in response to the complex and computationally demanding alternatives of traditional coding theory. It is an iterative protocol with the number of iterations decided beforehand based on the expected number of errors in the string. In each iteration Alice and Bob alternately transmit authenticated parity information on their respective strings back and forth to each other across the public channel in order to locate and correct the errors in Bob's string.

The protocol starts with Alice possessing the correct string  $\hat{\mathbf{x}}$ , and Bob the faulty string  $\hat{\mathbf{y}}$ , where  $n$  is the length. The first pass differs from the rest and starts with Alice and Bob agreeing on a number  $k_1$  and then dividing each of their

strings into  $\lceil \frac{n}{k_1} \rceil$  blocks with approximately  $k_1$  bits in each block. The blocks are numerated from  $v = 1 \dots \lceil \frac{n}{k_1} \rceil$  and the set  $K_v^u$  designates the bits in block number  $v$  in iteration number  $u$ .

Alice then proceeds to transmit the parity of each block. In the blocks where Alice's and Bob's parity do not match, an odd number of errors have occurred, and they execute a *BINARY protocol* in order to locate one erroneous bit. This entails Alice sending the parity of the first half of her block to Bob who compares it with the parity of his half and decides in which of the two halves there is an error. He conveys this to Alice who now considers this half as the new block, divides it in two and transmits the parity of the first half. This is repeated until Bob knows the exact location of the bit and then corrects it. Now each of Bob's blocks,  $K_v^1$ , has the same parity as Alice's and must therefore contain an even number of errors, possibly zero.

For each new iteration,  $u$ , Alice and Bob decide upon a number  $k_u$  as well as a random function to construct the  $\lceil \frac{n}{k_u} \rceil$  blocks, and Alice sends the parities of each block. As in the first pass, Bob and Alice execute a BINARY protocol in the blocks where their parities do not match. If Bob finds and corrects bit  $\hat{y}_i$  in some block  $K_v^u$ , he simultaneously disrupts all the blocks from the previous iterations that contained  $\hat{y}_i$ , which prior to the correction all had the same parity as Alice. The set of all these blocks are defined as  $X$ .

$$X = \{K_v^{\tilde{u}} \mid \tilde{u} < u, \hat{y}_i \in K_v^{\tilde{u}}\} \quad (3.3)$$

Another BINARY search is performed in the smallest block in  $X$ . Bob finds and corrects a new bit  $\hat{y}_j$ . All the blocks from iterations up to and including  $u$  that have contained  $\hat{y}_j$  comprise the set  $Y$ .

$$Y = \{K_v^{\tilde{u}} \mid \tilde{u} \leq u, \hat{y}_j \in K_v^{\tilde{u}}\} \quad (3.4)$$

The set,  $Z$ , of blocks that at this point contain an odd number of errors can be expressed in the following way:

$$Z = (X \cup Y) \setminus (X \cap Y), \quad (3.5)$$

since the bits in  $(X \cap Y)$  have been altered twice, and therefore do not disturb the parity of the blocks they are a part of.

The process of finding two bits,  $\hat{y}_i$  and  $\hat{y}_j$ , and correcting them in the way described above is continued until  $Z \neq \emptyset$ , which means that all Bob's blocks  $K_v^{\tilde{u}}$  for  $1 \leq \tilde{u} \leq u$  have matching parities with Alice's and thus have an even number of errors.

The main disadvantage of the CASCADE-protocol and others like it is that their interactive nature causes a big runtime overhead on the public channel.

Hence, the bulk of variations on CASCADE, such as the WINNOW-protocol[28] and the contribution of Liu[29], attempt to minimize the interactivity of the protocol at the expense of reconciliation efficiency. Nonetheless, two-way protocols seem to be the most popular choice in real-world implementations due to their low complexity and reasonably good information rates.

At the time of its discovery CASCADE outperformed all one-way reconciliation schemes in regards to information rate, and modifications have later been made that optimize the information rate even further[30, 31]. However, since the original version of CASCADE is the most widely implemented, it is the natural benchmark for comparing the performance of one-way error-correction protocols, which is subject to further study in Section 5.

### 3.1.5 Privacy amplification

Following a successful reconciliation scheme, Alice and Bob have identical keys,  $\hat{\mathbf{x}} = \hat{\mathbf{y}}$ , of size  $n$ . Eve can at this point have mutual information with the key from two different sources: active tampering on the quantum channel and passive eavesdropping on the public channel during the reconciliation scheme. This information needs to be removed through the process of *privacy amplification*, first introduced in 1988[26] and generally proven secure in 1995[32].

In essence it entails distilling a secret key,  $\hat{\mathbf{z}}$ , from  $\hat{\mathbf{x}}$  with which Eve has an arbitrarily low amount of mutual information, by sacrificing a certain amount of bits so that;  $|\hat{\mathbf{z}}| = k < |\hat{\mathbf{x}}| = n$ . Also this scheme can be performed using the universal<sub>2</sub> functions from Section 2.1.5. If Eve is in possession of a key,  $\tilde{\mathbf{x}}$ , which is correlated to  $\hat{\mathbf{x}}$ , the universal<sub>2</sub> hashing functions ensures that the probability of  $\tilde{\mathbf{z}} = \hat{\mathbf{z}}$  is less than or equal to  $2^{-k}$ , which leaves Eve's knowledge on the secret key no better than a random guess. Such a scheme can easily be implemented using binary *Toeplitz* matrices[33]

The most general theoretical attack Eve can perform is *probing* the qubit states without disturbing them, hopefully gaining some correlation with the qubits.

$$U(\rho_{AB} \otimes |0\rangle\langle 0|_E) = \rho_{AB} \otimes \rho_E(\rho_{AB}) \quad (3.6)$$

When the true bases are made public Eve can optimize her measurement strategies on  $\rho_E(\rho_{AB})$  gaining a maximal amount of information. She could also wait until after the error-correction protocol is performed and use the publicly available parity information to further enhance her measurement strategy. For that reason, it can be argued that Eve can gain more information on the key by exploiting the classical parity information, than the actual amount of bits passed over the public channel. Employing privacy amplification and shortening the key corresponding to the amount revealed during error-correction after such an attack might not remove all the information Eve gained from this amount.

One way to counter this threat is to encrypt all the parity information made public in error-correction with previously established key using OTP. This will reduce the net key gain with  $m = \kappa(p)h(p)$ . Obviously, Alice and Bob would then have to start out with a pre-established key with which they could encrypt the parity information in the first iteration. Given that such a restriction is already needed due to authentication, increasing the length of this starting key is not a profound modification.

Using OTP to encrypt the parity information means that the only mutual information Eve has with the key is through the active eavesdropping on the quantum channel. Compared to the amount of mutual information Bob has with Alice in Eq.(3.2), the amount possible for Eve to achieve is the complementary quantity,  $nh(p)$ . Also privacy amplification protocols are subject to finite-size degradations of the performance, such as  $\kappa(p)$  for the error-correction, but in this text perfect privacy amplification is assumed. Hence, using a universal<sub>2</sub> scheme as described above with  $k \leq n(1 - h(p))$ , gives Eve no better knowledge of the final key than a random guess.

In order to do the actual privacy amplification, Alice and Bob need to agree upon a specific universal<sub>2</sub> function. In the context above, Eve could also here wait until the function is made public and optimize her measurements correspondingly. Even though it is proven[34] that this will not give Eve any advantage, it is more convenient for the pair to use already established secret key to decide the function. Since any public agreement must be authenticated, the loss of secret key will not be substantially increased, while the time-overhead on the public channel will be decreased.

After both error-correction and privacy amplification the remaining key is then described by the *secret key rate*,  $s$ . Given that privacy amplification reduces the block from  $n$  to  $k$ , and that error-correction consumes  $m$  bits of key, the secret key rate in Eq. (3.9) describes the net key gain from each iteration of the QKD-protocol. The inequality comes from the fact that  $k \leq n(1 - h(p))$ , as well as the extra amount of key used in authenticating the communication between the two parties.

$$s < k - m \tag{3.7}$$

$$s < n(1 - h(p)) - n(\kappa(p) - h(p)) \tag{3.8}$$

$$s < n(1 - h(p))(1 + \kappa(p)) \tag{3.9}$$

If  $s$  is required to be positive, an upper bound can be found for  $p$ :

$$0 < s < n(1 - h(p)(1 + \kappa(p))) \quad (3.10)$$

$$0 < 1 - h(p)(1 + \kappa(p)) \quad (3.11)$$

$$1 > h(p)(1 + \kappa(p)) > 2h(p) \quad (3.12)$$

$$h(p) < \frac{1}{2} \quad (3.13)$$

$$p < 0.11. \quad (3.14)$$

Thus, to keep the secret key rate positive the maximum tolerable transition probability is 11%. Both the maximum limit on  $p$  and the secret key rate are however directly affected by the efficiency,  $\kappa$ , of the error-correction protocol.

## 3.2 Realization

Even though the theoretical description of a quantum key distribution scheme is very promising, the translation of those promises into real-world experiments is far from trivial. Nonetheless, QKD schemes are among the first realizations of quantum information theory and experience much experimental activity. It has even spawned commercial companies[35, 36] distributing optical setups that perform quantum key distribution. The security of these implementations are constantly challenged in order to develop the most secure setup possible.

This section will only mention some of the difficulties facing experimentalists within this field, with the intention of introducing the various elements which can severely influence the raw key-rate. Since a full study on this field is far out of the scope of this text, the reader is referred to the review articles[25, 27, 37] and references therein for a more comprehensive examination. The main emphasis will be on aspects directly related to the implementation of the BB84-protocol.

As previously mentioned, QKD is without exception performed using photons as qubits. One categorization of QKD-protocols is the utilization of discrete or continuous variables, the latter promising slightly higher raw-key rates using faster sources and homodyne detection schemes. Nonetheless, discrete variable QKD has been, and still is, the most widely implemented QKD systems. Discrete-variable protocols are usually sub-divided in two main branches, *entangled state protocols* like the Ekert-protocol[22] and P&M-protocols like the BB84[6] and B92[21] protocols. All of which theoretically assume perfect single photons sources to prepare the qubits and perfect single-photon detectors for the measurements.

### 3.2.1 Source

For QKD-protocols utilizing entangled states as qubits, spontaneous parametric down conversion is usually employed. This entails sending the qubit-photon

through an optical crystal with which it interacts in a non-linear way, resulting in two entangled photons with individually lower frequencies but with the degree of freedom used for encoding intact. These two photons are then distributed between Alice and Bob who can exploit their entangled nature to distill a key.

For discrete-variable protocols, it is essential for the photon source to approximate a single-photon source as closely as possible. This can be achieved using a heavily attenuated laser, which produces the following coherent state:

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle, \quad (3.15)$$

where  $\alpha$  is a physical parameter of the laser, and  $|n\rangle$  is the  $n$ -photon state. This gives a *Poissonian distribution* of the number states:

$$P(n) = \frac{(|\alpha|^2)^n}{n!} e^{-|\alpha|^2}, \quad (3.16)$$

with the expected number of photons equal to  $|\alpha|^2$ .

Even though the parameter  $\alpha$  of the laser is attenuated so low as to give the expected number of photons equal to one, there is still a non-zero probability of emitting more than one photon. This opens a possible security loophole for Eve, called the *photon number splitting (PNS)*-attack. She can in theory perform a *non-demolition* measurement on the coherent state,  $|\alpha\rangle$ , learning the actual number of photons in the laser pulse without disturbing the information of the qubits. If there is more than one photon present she can keep one of them, and after the information on the correct bases has been made public, she can measure it in the correct basis and gain full information on that one bit. This would not incur any detectable perturbation in Alice's or Bob's states. If she also suppresses single-photon pulses she could completely break the security of the BB84-protocol.

The first intuitive response from the QKD environment to this threat was to lower  $\alpha$  enough to keep the probability of multi-photon emissions arbitrarily low. Thanks to the recent discovery of the *Decoy protocol*[38], which has been proven secure[39], the BB84-protocol can be run with  $|\alpha|^2 \sim O(1)$ . With the decoy protocol Alice sends a set of "decoy-qubits" to Bob that are only used for the detection of Eve and not key extraction. By keeping all parameters equal to the normal "key-qubits" but only increasing  $\alpha$ , the decoys are indistinguishable from Eve's point of view. If she attempts a PNS-attack however, the decoy states will generally have a higher probability of reaching Bob than regular states since they are more often suppressed. This difference can be detected by Alice and Bob and the protocol can be aborted.

Nevertheless, values of  $\alpha$  in these low regimes still entails a significant probability of the laser pulse containing no photons.

### 3.2.2 Channel

Since photons are the qubits of choice, any media transmitting a photon can act as the quantum channel between Alice and Bob. In the lab, the preferred channel is usually optical fibers but *free space QKD* is also an active field of research.

The two main difficulties for a quantum channel are *loss* and *decoherence*. Optical fibers are very well documented with given rates of loss dependent on the distance and wavelength. Two "windows" of lowered losses, 1300nm and 1550nm, are dubbed the *telecom wavelengths* and are preferable for long-distance transmission. Due to the no-cloning theorem, qubits cannot be amplified, thus preventing QKD from being performed over arbitrarily long distances. In free space, losses are attributed both to the geometric difficulties of aligning the source and detector with a free line of sight, in addition to atmospheric disturbances where photons interact with particles in the Earth's atmosphere. In the futuristic concept of QKD between satellites, the atmospheric losses would be negligible.

One obstacle using optical fibers is *polarization mode dispersion*. This birefringence effect may depolarize the photon and makes polarization-based QKD unsuitable for optical fibers. Consequently, the degree of freedom most commonly used to implement a qubit with optical fibers, is phase. By splitting the pulse, Alice can introduce a relative phase between the two parts, which Bob needs to match in order to get deterministic measurements.

For free-space implementations this source of quantum decoherence is not a problem, leaving polarization a viable and ultimately preferable choice of encoding.

### 3.2.3 Detectors

In discrete-variable protocols the detection scheme on Bob's side is dependent on *single photon detectors (SPDs)*. This may seem like a simple principle, but turns out to be very hard to realize. The most commonly used SPDs in fiber optics implementations are so-called *avalanche photodiodes (APDs)*, which are either based on silicon (Si) or indium gallium arsenide (InGaAs). They utilize the photoelectric effect so that the impact of a photon causes an avalanche of electrons, which in turn results in a detectable current.

An APD has several characteristics that determine its performance, but these are notoriously hard to optimize simultaneously.

- It needs to have a high efficiency,  $\eta$ , meaning that when hit by a single photon it has a high probability of detecting it.
- The *dark count rate*,  $\sigma$ , indicates how often it reports a detection without being hit. A parameter that obviously needs to be minimized.

- It also needs a high operating frequency,  $\nu$ , which is important in order to keep a high key-rate.

The two SPDs mentioned above, the Si- and InGaAs-APDs, have their different strengths and weaknesses. While the Si-APD can have  $\eta > 70\%$ ,  $\sigma \approx 50\text{Hz}$ , and can work close to room-temperature ( $-20^\circ\text{C}$  to  $-10^\circ\text{C}$ )[37], their wavelength-efficiency characteristics make them impractical for use in QKD. Their peak efficiency is at 800nm, which is perfect for visible light, but at the telecom wavelengths used in fiber optics they are highly inefficient. InGaAs-APDs on the other hand are able to detect photons with telecom wavelength, but only with an efficiency  $\sim 10\%$ . Additionally, they have to be cooled to between  $-50^\circ\text{C}$  and  $-100^\circ\text{C}$  and still exhibit significant dark count rates,  $\sigma \sim 10^{-5}$  per *gate*[27].

After a successful detection, SPDs are very prone to dark counts, called the *after-pulse effect*. In order to deal with this, the detectors have an intrinsic *dead-time* after a detection in which they cannot fire. This severely limits the operating frequency. An efficient way of minimizing the effect of dark counts is to *gate* the detector. This means allowing it to fire only in a time interval expected to contain a photon. With a suitable gating scheme, reports have been made of InGaAs-APDs able to operate with  $\nu = 500\text{MHz}$ .

The factors  $\eta$  and  $\nu$  of the detector along with the pulse frequency of the laser source and its probability  $P(1)$ , largely decide the possible raw-key rate possible for a discrete-value QKD-protocol. This is however a highly active research field with constant developments.

In discrete-value protocols, at least two detectors are normally used. One for each qubit value. Due to the imperfections of real-world implementations, Bob may experience a *detector efficiency mismatch (DEM)*, where the efficiency of the two detectors do not match perfectly. If the detector efficiency is skewed in the time domain, possibly caused by non-synchronous gating, Eve could exploit this by employing a *faked-state attack*[40]. After intercepting and measuring the original qubit from Alice, she could transmit a faked state to Bob and time it so that it arrives when one of the detectors has a higher efficiency than the other. Assume that the same detector is used for the states  $\{|0\rangle, |+\rangle\}$ , and the other for  $\{|1\rangle, |-\rangle\}$ . If Eve then wants to impose the state  $|0\rangle$  on Bob she could prepare the state  $|-\rangle$  and time the arrival to when the first detector is more efficient. In the cases where Bob chooses to measure in the X-basis he will always measure  $|-\rangle$ , but with a lower probability due to the lowered efficiency of the second detector. In the opposite case the measurement outcome is theoretically random, but due to the DEM Bob will have a lower probability of measuring  $|1\rangle$ . This leads to a detectable decline in the raw-key rate from Alice's and Bob's point of view, which could theoretically be countered from Eve's side by switching to a more powerful qubit-source.

Another attack that exploits DEM is the *time-shift attack*[41] in which Eve

simply manipulates the time of arrival of the photon at Bob. Experimentalists faced with such loopholes have the option to try and apply a practical fix, which often makes the implementation more complex, or to try and quantify Eve's advantage and increase the privacy amplification accordingly. Both these security loopholes have been patched[42] at the expense of the secret key rate. A big challenge for the field at the present is to develop a security proof that can encompass all possible real-world component imperfections.

### 3.3 Security

The ability to prove the unconditional security of QKD is the founding thought of the entire field. However, to do so is highly non-trivial and many different approaches have been attempted. Due to the diverse and prolific nature of this field, only a few proofs will be discussed.

#### 3.3.1 Definition

When discussing security in this setting it is of vital importance to have an unambiguous definition of what it means to have a secure protocol. It was not until 2005 that Renner [43] introduced a novel way of mathematically defining an  $\epsilon$ -secure system.

$$\frac{1}{2} \|\rho_{AE} - \rho_A \otimes \rho_E\| < \epsilon \quad (3.17)$$

The state to the left in this definition,  $\rho_{AE}$ , represents a general state produced by a realistic system.  $\rho_A \otimes \rho_E$  on the other hand, is the ideal case where Eve's state,  $\rho_E$ , is completely uncorrelated with Alice's random key,  $\rho_A$ . If the distance between the actual state and the ideal state is less than  $\epsilon$ , the system is regarded as  $\epsilon$ -secure. The distance  $\|\rho\|$  in this case is defined as  $\text{Tr}(\sqrt{\rho^\dagger \rho})$ .

The main advantage of this new security definition is that it is *composable*. When a scheme is  $\epsilon_1$  secure it means it has a failure probability of maximum  $\epsilon_1$ . So if a key is  $\epsilon_1$ -secure and subsequently used in an  $\epsilon_2$ -secure scheme of some sort, the total probability of failure is  $\leq \epsilon_1 + \epsilon_2$ . Since the established key after one round of the BB84 is used to authenticate information in the subsequent rounds, the failure probability for each round needs to be  $\ll \frac{1}{N}$  if  $N$  is the number of rounds.

#### 3.3.2 Proofs

What all security proofs have in common is the explicit or implicit connection between the security condition of Eq. (3.17) and the length,  $k$ , of the resulting secret key,  $\hat{\mathbf{z}}$ . Given the unconditional security of the privacy amplification

protocol[32, 34], the main difficulty is quantifying the amount of privacy amplification needed to be done in order to satisfy the security requirement.

One of the most well-known security proofs within this field is the milestone paper by Shor and Preskill[23], which is a refined and simplified version of the early contribution of Lo and Chau[44]. By showing that entanglement is a sufficient condition for security and the equivalence between the Ekert- and the BB84-protocol, they were able to prove unconditional security. They managed this through assuming that Alice and Bob were capable of performing local quantum computations to perform quantum error-correction and privacy amplification with so-called *Calderbank-Shor-Steane (CSS)*-codes. In that way they could distill perfectly entangled photon-pairs, which could easily be proven secure. This approach puts severe restrictions on the practical implementation of error-correction and privacy amplification since they have to collectively correspond to a quantum CSS-code in order for the security proof to hold. Because of this enforced connection between the two protocols it has been impossible to separately study good or bad alternatives for them. Even though progress has been made in decoupling the analysis of the two by encrypting the parity information from the error-correction part[45], they are still subject to restrictions from the CSS-codes.

This limitation sparked a new interest in the early security proof of Mayers from 1996[46]. His approach was through the application of the uncertainty principle, which gives the proof the nice feature of keeping the error-correction and privacy amplification protocols decoupled. The complexity of the original proof was alleviated by the work of Koashi and Preskill[47] at the expense of introducing a strict symmetry requirement to the BB84-protocol.

The recent paper of Koashi[48] uses notions of complementarity to prove security while maintaining the advantages of both the previous schools of thought. Since error-correction is decoupled from privacy amplification one can simply find the best protocol possible, independent of the other, with guaranteed security.

Koashi's proof assumes Alice and Bob use an entanglement based protocol, in which case they share  $n$  *Bell states*:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} = \frac{|+\rangle + |-\rangle}{\sqrt{2}}, \quad (3.18)$$

where the first qubit is sent to Alice and the second to Bob. The basis Alice chooses to measure in is equivalent to the preparation basis of Alice in a P&M-protocol, which means that Alice in both cases is considered to be holding the correct key. This is true because when Alice measures in one of the bases, the bell state collapses into the state corresponding to the measurement outcome. Alice and Bob are assumed to always measure in the Z-basis for key-generation, and occasionally measure in the X-basis strictly for estimation purposes. If the

estimation is neglected, the sifted key can be regarded as the outcome of only Z-basis measurements, with the basis-dependent transition probability,  $p_Z$ .

Koashi proceeds to suggest a virtual protocol where only the X-basis is used, with corresponding transition probability,  $p_X$ . The problem is then to evaluate how well the outcome of such a measurement could be assessed. If Bob gets the measurement outcome  $\zeta_B^X$ , the remaining uncertainty on Alice's outcome,  $\zeta_A^X$ , is upper bounded according to  $p_X$ :  $H(\zeta_A^X|\zeta_B^X) \leq nh(p_X)$ . Since the outcome of the measurement in a complementary basis to Z can be predicted to such a high degree of accuracy, the outcome of the Z basis measurement must be hard to predict according to the uncertainty principle. This difficulty is not only applicable to Bob, but to anyone attempting to assess the Z-basis measurement outcome, including Eve. Formally, the entropic uncertainty relation[49] states that  $H(\zeta_A^Z) + H(\zeta_A^X) \geq n$ , which means that the uncertainty anyone has on the Z-basis outcome is given by:  $H(\zeta_A^Z) \geq n - H(\zeta_A^X) = n - nh(p_X) = n(1 - h(p_X))$ . This is the amount of secret key Alice is able to distill from the qubits. Considering the extra amount of parity information Bob needs in order to correct his key,  $n\kappa(p_Z)h(p_Z)$ , the secret key rate reads as:  $s < n(1 - h(p_X) - \kappa(p_Z)h(p_Z))$ . Consequently, the transition probability in the X-basis determines the amount of privacy amplification, while the transition probability in the Z-basis determines the error-correction. While such protocols do exist, the adaptation of this mindset to the conventional implementation of the BB84-protocol with randomly chosen bases is as follows. The transition probability  $p_Z$  relates to the transition probability in the basis that is actually used, while  $p_X$  gives the abstract notion of the error-rate in the conjugate basis to the one which at any time is used. While the true value of  $p_X$  can never be known in real implementations, it suffices to use the maximum of the two probabilities actually measured for both the X- and Z-bases.



## 4 Classical Coding Theory

### 4.1 Linear Codes

The following presentation of basic coding theory follows the introductory textbook of Huffman and Pless[50] as well as the review article of Johnson[51]. For supplementary input the reader is asked to consult these texts.

A *code*  $C$  of *dimension*  $n$  is basically a set of assorted codewords of equal length,  $n$ . The most common restriction to place on a code is that of *linearity*, in which case the codewords constitute a linear subspace of dimension  $a$  of the  $n$ -dimensional space.

Each *linear code* can be represented by a *generator matrix*,  $\mathbf{G}$ , with dimensions  $a \times n$ , where the rows constitute a basis for the subspace  $C$ . The *code rate* for a linear code is then described as  $r = \frac{a}{n}$ , and is equivalent to the information rate introduced in Eq. (2.3). Since there are  $a$  basis-vectors and only two letters in our alphabet, there is a total of  $2^a$  codewords in such a code. In the same way that there can be many different bases for a subspace, a code can have many different generator matrices. However, if the first  $a$  columns of  $\mathbf{G}$  are linearly independent, the generator matrix can be written in *standard form*, which is unique for each code. Standard form implies that the first  $a$  columns in  $\mathbf{G}$  form the identity matrix.

$$\mathbf{G} = [\mathbf{I}_a | \mathbf{W}] \quad (4.1)$$

Now a message,  $\hat{\mathbf{m}}$ , of length  $a$  can be encoded into a codeword in  $C$  by means of a simple matrix multiplication,  $\hat{\mathbf{m}}\mathbf{G} = \hat{\mathbf{c}}$ . Encoding the message  $\hat{\mathbf{m}} = [101]$  using the  $3 \times 6$  linear code below gives the codeword  $\hat{\mathbf{c}} = [101110]$  as follows.

$$\underbrace{[1 \ 0 \ 1]}_{\hat{\mathbf{m}}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{G}} = \underbrace{[1 \ 0 \ 1 \ 1 \ 1 \ 0]}_{\hat{\mathbf{c}}} \quad (4.2)$$

Notice that having the generator matrix in standard form returns the message in the first  $a$  bits of the codeword. This clearly illustrates the difference between *information carrying bits* and the deliberately introduced *redundancy bits* in a codeword. Even though the last 3 bits are described as redundant, they are of critical importance in the effort to detect and correct any errors the codeword might be subject to through the channel.

An alternate, but equivalent representation of a code,  $C$ , is the *parity-check matrix*,  $\mathbf{H}$ . It is defined such that  $\mathbf{H}\hat{\mathbf{x}}^T = 0$  if and only if  $\hat{\mathbf{x}} \in C$ . If  $\mathbf{G}$  is in standard form then the parity-check matrix can be written in the following way:

$$\mathbf{H} = [-\mathbf{W}^T | \mathbf{I}_{n-a}]. \quad (4.3)$$

This is obvious since  $\mathbf{H}\mathbf{G}^T = -\mathbf{W}^T \oplus \mathbf{W}^T = \mathbf{0}$

When Alice transmits  $\hat{\mathbf{c}}$  over a noisy channel, Bob receives  $\hat{\mathbf{y}} = \hat{\mathbf{c}} \oplus \hat{\mathbf{e}}$ , with  $\hat{\mathbf{e}}$  being the *error-vector*. Now Bob can check and see if  $\hat{\mathbf{y}} \in C$  by calculating the *syndrome*,  $\hat{\mathbf{s}}$ .

$$\mathbf{H}\hat{\mathbf{y}}^T = \mathbf{H}(\hat{\mathbf{c}} \oplus \hat{\mathbf{e}})^T = \mathbf{H}\hat{\mathbf{e}}^T = \hat{\mathbf{s}}^T \quad (4.4)$$

The rows in  $\mathbf{H}$  act as checksums so if  $\hat{\mathbf{s}} \neq \hat{\mathbf{0}}$  then Bob knows  $\hat{\mathbf{e}} \neq \hat{\mathbf{0}}$ , which means that an error has occurred. Otherwise, the received vector is a codeword and thus deemed uncorrupted during the transmission.

Continuing with the example above, if  $\hat{\mathbf{c}}$  is subject to the error-vector  $\hat{\mathbf{e}} = [100000]$ , Bob receives  $\hat{\mathbf{y}} = [001110]$  and observes the syndrome  $\hat{\mathbf{s}} = [101]$ .

$$\underbrace{\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}}_{\hat{\mathbf{y}}^T} = \underbrace{\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}}_{\hat{\mathbf{s}}^T} \quad (4.5)$$

The parity-check matrix  $\mathbf{H}$  is deduced from  $\mathbf{G}$  via Eq. (4.3). Even though there apparently has occurred one or more errors here, there is no intuitive way of immediately deducing which bits are faulty directly from the syndrome.

Coding theory always operates on the assumption that the fewest possible errors has occurred. That means finding the error-vector  $\hat{\mathbf{e}}$  with lowest weight, which results in the observed syndrome according to Eq. 4.4. Combining that with the received vector,  $\hat{\mathbf{y}}$ , gives Bob's best guess for  $\hat{\mathbf{c}}$ . This is called *maximum likelihood* (ML) decoding and is heavily dependent on the *minimum distance* characteristics of the code. The minimum distance is defined as the lowest distance between any two codewords in the code and is denoted  $d_{\min}$ . If the number of errors is less than  $d_{\min}$ , that is if  $\text{wt}(\hat{\mathbf{e}}) < d_{\min}$ , the presence of errors is detectable since there are too few errors to convert the original codeword,  $\hat{\mathbf{c}}$ , into another valid codeword,  $\tilde{\mathbf{c}}$ . It is however not necessarily decodable since if  $\text{wt}(\hat{\mathbf{e}}) > \frac{d_{\min}}{2}$ ,  $\hat{\mathbf{y}}$  might be closer to  $\tilde{\mathbf{c}}$  than  $\hat{\mathbf{c}}$  and hence decoded as  $\tilde{\mathbf{c}}$  using ML decoding. If  $\text{wt}(\hat{\mathbf{e}}) < \lfloor \frac{d_{\min}-1}{2} \rfloor$  however, ML decoding will always be able to decode  $\hat{\mathbf{y}}$ , as  $\hat{\mathbf{c}}$  invariably will be the closest codeword.

Theorem 1.4.2 in Huffman and Pless[50] states that for a linear code,  $d_{\min}$  is equal to the minimum weight of all the non-zero codewords in the code.

$$d_{\min} = \min_{\hat{\mathbf{c}} \in C} \text{wt}(\hat{\mathbf{c}}), \quad (4.6)$$

The code in the above example has  $2^3 = 8$  codewords, so a simple evaluation gives  $d_{\min} = 3$ . Hence it can detect 2 errors, and correct 1. By using ML decoding on  $\hat{\mathbf{y}}$  in the example, Bob would easily find that error-vector  $\hat{\mathbf{e}} = [100000]$  is the only error-vector with weight 1 that results in codeword when combined with  $\hat{\mathbf{y}}$ , and thus deduce that  $\hat{\mathbf{c}}$  was the sent codeword. If  $\hat{\mathbf{c}}$  had been subject to the error-vector  $\hat{\mathbf{e}} = [110000]$ , the resultant vector would be  $\hat{\mathbf{y}} = [011110]$ . This vector would also produce a non-zero syndrome,  $\hat{\mathbf{s}} = [001]$ , but now the closest codeword to  $\hat{\mathbf{y}}$  is  $\tilde{\mathbf{c}} = [010110]$  and ML decoding would fail.

Even though  $d_{\min}$  was easily found for this specific code, it is generally a highly non-trivial exercise for larger-sized codes. As a consequence, highly structured algebraic codes has been the most widely used, as they are better suited for theoretical optimization of parameters such as  $d_{\min}$ . Smaller sized codes are also convenient in light of ML decoding.

Such a brute force exhaustive search through  $2^a$  codewords obviously becomes extremely inefficient for realistic code-sizes. Hence, by cleverly constructing  $\mathbf{G}$  and  $\mathbf{H}$ , many different protocols of error-correction have been designed within this framework, spanning from the relatively simple *Reed-Solomon codes*[52] to the new concept of *serially concatenated convolutional codes*, also called *Turbo codes*[53].

## 4.2 Low-Density Parity-Check Codes

As the name implies, *Low-Density Parity-Check(LDPC) codes*, are codes where the parity-check matrices have very low densities, meaning a low ratio of non-zero entries. As we operate in the binary realm, the non-zero entries are all ones.

LDPC codes were first described by Gallager in his phd-thesis in 1963[8], and later rediscovered by MacKay in 1999[7]. The ingenuity of these codes does not lie in their mere sparsity but rather on the iterative algorithms with which they are decoded. In fact, the same decoding algorithms can successfully be applied with conventional linear codes, but the sparsity of the LDPC codes ensures a feasible linear increase in complexity as  $n$  grows, which will be discussed in Section 4.4.3.

As mentioned in the previous section, the minimum distance of a code is a very important parameter for a code, deciding the limit for decodability. The problem of finding the minimum distance for a specific code however, is a problem that has proven to be unsolvable in polynomial time[54]. Even though it is analytically infeasible, there exists numerical algorithms[55, 56] that attempt to find  $d_{\min}$  for LDPC codes, but they are unstable and impractical for large block-sizes. Since the search for the minimum distance in most cases entail finding the range of decodability for a code, a simpler approach is to perform simulations and deduce the minimum distance from the decodable range of errors.

### 4.3 Construction

The construction of LDPC codes is usually not done in the conventional way of first devising  $\mathbf{G}$  and then attaining  $\mathbf{H}$  through Eq. (4.3). The reason for this is firstly because it might be demanding to get  $\mathbf{G}$  in standard form, and secondly because it is difficult to deduce the attributes of  $\mathbf{H}$  solely on the basis of  $\mathbf{G}$ : a sparse generator matrix does not necessarily produce a sparse parity-check matrix. For those reasons, LDPC codes are usually designed the other way around, so henceforth code-construction will imply constructing the parity-check matrix of the code.

#### 4.3.1 Graphs

A convenient way of representing LDPC codes, in addition to other binary codes, is through *graphs*, since it provides an intuitive way of visualizing the construction of the code as well as the decoding algorithms.

Graphs in this context consists of different types of *nodes*, connected by *edges*. In the specific case when there are only two types of nodes and the edges exclusively connect nodes of different types, the graph is called a *bipartite graph*. As first described by Tanner[57], bipartite graphs are a superb way of visualizing linear codes. Hence the term *Tanner-graphs* is often utilized in such instances.

Each column in  $\mathbf{H}$  corresponds to a bit, or more generally a *variable*, in the received vector  $\hat{\mathbf{y}}$ . In a Tanner-graph each column constitutes a node, which are conventionally visualized as circular *nodes*. Equivalently, each row in  $\mathbf{H}$  represents a *check* that needs to be fulfilled, and will be depicted as a square *node*. So if  $\mathbf{H}$  is of dimensions  $m \times n$ , there are  $n$  variable nodes and  $m$  check nodes in the graph. For  $\mathbf{H}_{i,j} \neq 0$ , there is an *edge* connecting check-node  $i$  and variable-node  $j$ . The number of edges emanating from a node is called the *degree* of the node.

A matrix that produces a graph is called the *incident matrix* of a graph. Since a parity-check matrix produces a unique Tanner-graph, and a Tanner-graph has a unique incident matrix, the terms *code*, *graph* and *parity-check matrix* will be used interchangeably. As the graph representation is a more convenient representation for both construction and decoding, it will be most frequently used.

The parity-check matrix used in the preceding example, Eq. (4.5), spawns the Tanner-graph illustrated in Fig. 4.1.

In a graph, a *cycle* is a path of *length*,  $l$ , from one node via  $l$  edges through  $l - 1$  different nodes that returns you to the node of origin. In Fig. 4.1 a cycle of length 6 is present;  $(v_1 \rightarrow c_1 \rightarrow v_2 \rightarrow c_2 \rightarrow v_3 \rightarrow c_3 \rightarrow v_1)$ , since there are 6 edges connecting  $v_1$  with itself.

The *girth*,  $g$ , of a code is defined as the shortest cycle in a graph. By first evaluating the smallest cycle connecting each variable node with itself,  $g_i$ , one can

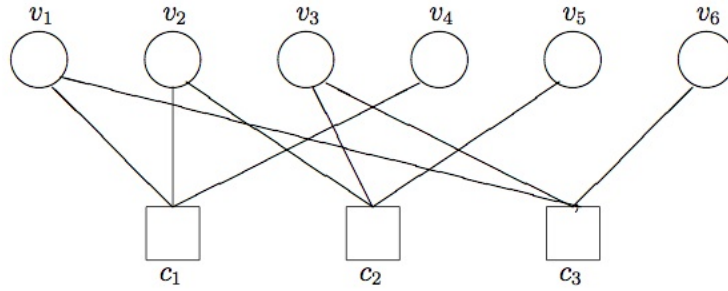


Figure 4.1: Tanner-graph

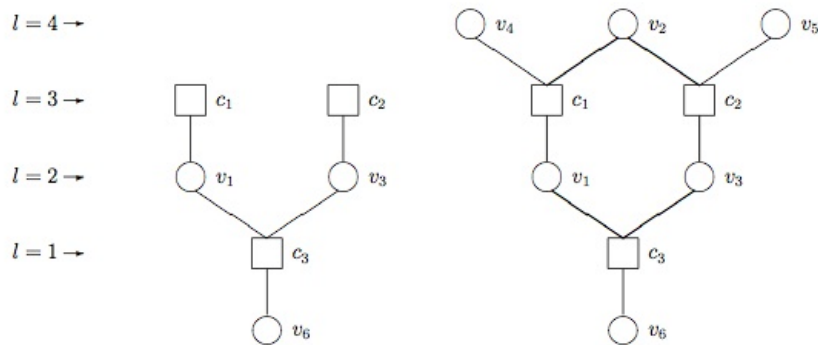


Figure 4.2: Tree-expansion

find the total girth of a graph by taking the minimum of those;  $g = \min_i g_i$ . In the example nodes  $g_1 = g_2 = g_3 = 6$ , as they are all in the same cycle and there are no shorter cycles connecting them to themselves. The last three nodes have no cycles, thus the girth of the Tanner-graph described by  $\mathbf{H}$  is 6.

If one is able to expand  $l$  steps from one node along all edges and not encounter any cycles, the resulting expansion is called a *tree of depth  $l$* . As can be seen from Fig. 4.2, a tree of depth 3 can be expanded from node  $v_6$ , while if the depth is increased to 4, the cycle described above becomes incorporated and the expansion is no longer a tree. Both the notions of cycles and trees are important in the construction- and decoding algorithms used with LDPC-codes.

For a more detailed approach to graph-theory than what is given here, the reader is asked to refer to the text book by Bollobas[58].

### 4.3.2 Distributions

In their first conception, LDPC codes were described by Gallager[8] as having a fixed number of non-zero entries in each row,  $d_c$ , and each column,  $d_v$ , for any given code,  $\mathbf{H}$ , typically denoted an  $(n, d_v, d_c)$ -code. The subscripts  $c$  and  $v$  in this case signifies the check and variable nodes in a code, reflecting the equivalence between the two perspectives; each variable node has degree equal to  $d_v$ , thus influencing  $d_v$  parity-check equations, and each parity-check equation is a function of  $d_c$  variables, corresponding to the degree of the check nodes. In order to make the total number of non-zero entries unambiguous, the size of the code needs to fulfill this equation:

$$n \cdot d_v = m \cdot d_c. \quad (4.7)$$

Usually the code rate is decided beforehand, which means that by Eq. (2.3),  $m$  scales as  $(1 - r)n$ . This makes the size of  $\mathbf{H}$  grow quadratically with  $n$ , while the number of ones, through Eq. (4.7), only grows linearly, resulting in the density of  $\mathbf{H}$  being inversely proportional with  $n$ . This is an important feature of LDPC codes, which is often critical in analyzing ensembles of codes[7, 8].

By flipping Eq. (4.7) one gets  $\frac{m}{n} = \frac{d_v}{d_c}$ , which substituted into Eq. (2.3) gives:

$$r = \frac{n - m}{n} = 1 - \frac{m}{n} = 1 - \frac{d_v}{d_c}. \quad (4.8)$$

Codes that adhere to such restrictions are now referred to as *regular codes*: an expression that stems from *regular graphs* where within every group of nodes, each node has a set degree. In their modern resurrection however, the field of LDPC codes has been expanded to incorporate *irregular codes*. In the irregular case the degree of nodes of one type are no longer governed by a fixed number, but rather a *degree distribution*, as first introduced by Luby[59].

There are two equivalent ways of viewing the irregularity of a Tanner-graph. From the node point of view, the distributions  $b_i$  and  $r_i$  describes the fraction of nodes with degree  $i$  for variable- and check nodes respectively. This leads to the extension of Eq. (4.7):

$$n \cdot \left( \sum_{i=1}^{d_v^{\max}} b_i \cdot i \right) = m \times \left( \sum_{i=1}^{d_c^{\max}} r_i \cdot i \right), \quad (4.9)$$

where  $d_c^{\max}$  and  $d_v^{\max}$  denotes the maximum value of  $i$  with  $r_i \neq 0$  and  $b_i \neq 0$  respectively.

Rather than all check nodes having one fixed degree, e.g. six giving  $r_6 = 1$ , it is now possible to assign half the check nodes degree five and the other half seven,

giving the distribution  $r_5 = 0.5$ ,  $r_7 = 0.5$ . Consequently Eq. (4.8) can be extended accordingly:

$$r = \frac{n - m}{n} = 1 - \frac{m}{n} = 1 - \frac{\sum_{i=1}^{d_v^{\max}} b_i \cdot i}{\sum_{i=1}^{d_c^{\max}} r_i \cdot i}. \quad (4.10)$$

While this perspective is the most convenient in the practical construction of codes, another perspective is more commonly used in the literature. By using the edge point of view, the procedure described in Section 4.5 called density evolution, can be very concisely formulated[59]. The distributions  $\lambda_i$  and  $\rho_i$  signifies the fraction of the total number of edges that emanate from variable- and check nodes with degree  $i$ , respectively. For notational convenience they are traditionally incorporated into the expressions  $\lambda(x) = \sum_i \lambda_i \cdot x^{i-1}$  and  $\rho(x) = \sum_i \rho_i \cdot x^{i-1}$ . The use of  $x^{i-1}$  in these definitions allows an even more elegant expression for the rate:

$$r = 1 - \frac{\sum_{i=1}^{d_c^{\max}} \frac{\rho_i}{i}}{\sum_{i=1}^{d_v^{\max}} \frac{\lambda_i}{i}} = 1 - \frac{\int_0^{\infty} \rho(x) dx}{\int_0^{\infty} \lambda(x) dx}. \quad (4.11)$$

A (3,6)-regular code would yield the distributions  $\lambda(x) = x^2$  and  $\rho(x) = x^5$ .

These sums all start at  $i = 1$  since it makes no sense having a node with degree zero. They are regarded as probability distributions so they abide by the standard restrictions;  $0 \leq b_i \leq 1, \forall i$  and  $\sum_{i=1}^{d_v^{\max}} b_i = 1$ , equivalent for  $r_i, \lambda_i$  and  $\rho_i$ .

Since the optimization results from density evolution are normally given using  $\lambda(x)$  and  $\rho(x)$ , the following equations are frequently used for conversion:

$$b_i \cdot i = \frac{\lambda_i}{\sum_{j=1}^{d_v^{\max}} \frac{\lambda_j}{j}} \quad (4.12)$$

$$r_i \cdot i = \frac{\rho_i}{\sum_{j=1}^{d_c^{\max}} \frac{\rho_j}{j}} \quad (4.13)$$

By today's research there is still no analytical evidence as to why irregular codes are superior to regular ones, however, intuitive arguments have been presented as to why it must be the case[60]. Accordingly, there are still no analytical tools to decide upon the optimal weight distributions neither. Nevertheless, it is possible to place certain restrictions on the weight distributions and use linear programming techniques to optimize the remaining free variables, a process to be discussed further in Section 4.5.

### 4.3.3 Edge placement

The preceding sections has laid the foundation for the actual construction of the code by establishing the amount of nodes of each type, and the distributions of

their degrees. The actual placement of the edges, or placement of ones in the vast matrix of only zeros, is the next challenging task.

In Section 4.4.1 it will be explained why the presence of cycles in the code can, and will, reduce its decodability. Hence it is important to maximize the girth of the code, and especially prohibit cycles of short length. In fact, Gallager[8] gave an algorithm for constructing codes with good girth, but it is too inefficient for realistic block-sizes and only applies for regular codes.

Therefore, most early construction algorithms have been random[61], relying on the sparsity of the matrix to give good girth. Slightly more sophisticated methods include steps to ensure that no short cycles are present[7, 62, 63]. The method proposed by MacKay[7] for producing regular graphs consisted of randomly placing ones in a zero-matrix, column-by-column, following the restrictions on  $d_c$  and  $d_v$ . This was followed by a removal of columns, or variable nodes, which contributed to cycles of length four. Consequently, such codes has girth  $g \geq 6$ .

A more active tactic for building a Tanner-graph was suggested in 2001 called *progressive edge-growth*, or *PEG*[64]. With it, the authors provided a non-algebraic algorithm that progresses edge-by-edge, placing them in a way that minimizes the effect each edge has on the current girth.

In the initialization it is important to differentiate between target degree, as defined from the weight distributions  $\lambda(x)$  and  $\rho(x)$ , and the current degree a node experiences through the construction process. The variable and check nodes are initialized with increasing target degree, meaning target degree of  $v_i \leq$  target degree of  $v_j$  provided  $i \leq j$ .

The PEG algorithm iterates through the variable nodes, and executes the following algorithm for each edge:

- If it is the first edge to be established for variable node  $v_i$ , a random selection is made amongst the check nodes of smallest current degree. An edge is established between these two nodes.
- If it is not the first edge for variable node  $v_i$ , a tree expansion is made from this variable node out to depth  $l$ . The check nodes encompassed in this expansion constitute the set  $\mathcal{N}_{v_i}^l$  and the expansion progresses until  $|\mathcal{N}_{v_i}^{l+1}| = |\mathcal{N}_{v_i}^l|$  or  $\overline{\mathcal{N}_{v_i}^{l+1}} = \emptyset$ , with  $\overline{\mathcal{N}_{v_i}^{l+1}}$  being the complementary set to  $\mathcal{N}_{v_i}^l$ , so that  $\overline{\mathcal{N}_{v_i}^l} + \mathcal{N}_{v_i}^l$  constitutes the set of all check nodes. An edge is established between  $v_i$  and a random check node with lowest current degree, drawn from  $\overline{\mathcal{N}_{v_i}^l}$ .

This approach ensures that any cycle going through the current edge has length greater than  $l + 1$ . Constantly choosing random check nodes from the ones with

lowest current degree leads to a graph with a close to uniform check-node degree distribution. Hence, the PEG algorithm favors highly concentrated weight distributions. An expression for the lower bound[65] on the girth justifies this.

$$g \geq \lfloor t \rfloor + 1 \quad (4.14)$$

where

$$t = \frac{\log(m \cdot d_c^{\max} - \frac{m \cdot d_c^{\max}}{d_v^{\max}} - m + 1)}{\log((d_v^{\max} - 1) \cdot (d_c^{\max} - 1))} \quad (4.15)$$

A concentrated weight distribution minimizes  $d_c^{\max}$  and  $d_v^{\max}$ , thus maximizing  $g$ .

A slight modification of this scheme is called the *non-greedy* version. For large  $n$  it might not be necessary to expand the tree all the way in step two because the inherent sparsity of the graph will ensure good girth properties. By introducing a maximum depth to which the tree will expand,  $l_{max}$ , one can dictate the minimum girth of the graph and simultaneously significantly lower the construction time.

One of the main downsides of LDPC codes is the problem of encoding the messages. This process, as described in a basic fashion in Eq. (4.2), normally scales as  $n^2$ [62]. Naturally, a great deal of effort has been put into creating codes,  $\mathbf{H}$ , which spawn generator matrices that allow linear, or close to linear, time encoding[62, 59, 66]. Due to the practicalities described in Section 3.2 and embellished on in Section 5.1, such research will not be given much attention in this text.

## 4.4 Decoding

Since both the performance of the decoding algorithm in Section 4.4.1 and the security of quantum key distribution in Section 3.3 are given in the asymptotic limit of  $n \rightarrow \infty$ , a sufficient block-size is of paramount importance. However, the standard ML decoding scheme has to evaluate  $2^a = 2^{(n-m)} = 2^{(n-(1-r)n)} = (2^r)^n$ , which will quickly become vastly inefficient with realistic block-sizes.

Gallager already proposed an answer to this problem in 1963[8]. He introduced two iterative decoding algorithms that employ so-called *message-passing* between the nodes. In fact, it was the similarity of these algorithms to the iterative algorithms used to decode the, at that time, relatively new Turbo-codes, which sparked the renewed interest in LDPC codes in the 1990's.

The first one, which will be denoted *Gallager A*, passes *hard messages* meaning either 0 or 1. The parity checks are evaluated and the bit-values in variable nodes that appear in more than a fixed number of unsatisfied check nodes is flipped. The parity checks are reevaluated and if some of the check nodes are not satisfied the process is repeated until they all are.

In the language of message passing, the variable nodes first pass their hard information on what their bit-values are, 0 or 1, to the check nodes. They in turn sum the incoming messages up modulo 2 and pass the parity value back to the variable nodes. If the variable node receives more than a given amount of 1's from the check nodes it passes the opposite hard decision in the next iteration.

In the context of graph theory, the passing of messages can be regarded as spreading through a tree expansion from each variable node. In the first iteration each variable node is affected by the check nodes of depth 1, who in turn are affected by the variable nodes of depth 2. These variable nodes in depth 2 were in the same iteration and in the same way affected by the variable nodes of depth 4. Therefore in the next iteration, when the variable nodes receive updated messages from depth 2, these messages contain information from depth 4. In that way, each variable node is in iteration  $l$  influenced by a tree of depth  $2l$ .

#### 4.4.1 Belief propagation

The *Gallager A* algorithm is easily evaluated analytically and easily implemented, but is far from optimal. Thus he introduced a probabilistic modification to this scheme that incorporates the incoming information from the tree in a more systematic way. The algorithm, later dubbed *Belief Propagation*, was reinvented by MacKay[7] and has been the primary choice of decoding LDPC codes.

The messages in a message-passing protocol are defined to only contain *extrinsic information*. This means that the messages going out along one edge cannot be influenced by the information coming in along the same edge. Hence each node has to calculate a separate outgoing message for each edge, only based on the incoming messages from all the other edges. Cycles of length  $l$  in the graph will in the same way cause nodes to indirectly influence themselves after  $l$  iterations. Therefore, a graph with girth  $g$  will have  $g - 1$  optimal iterations before the messages become contaminated by their own information, encouraging the work on codes with high girths.

For that reason we define the following sets. For each check node  $i$ , the set  $\Omega(i)$  is the set of all variable nodes adjacent to check node  $i$ , while  $\Omega(i)\setminus j$  denotes the same set except for variable node  $j$ . In the same way, the set  $\Phi(j)$  is defined as the set consisting of all check nodes adjacent to the variable node  $j$ , with the same extension,  $\Phi(j)\setminus i$ .

The algorithm consists of two alternating steps in which the nodes update and pass *soft messages*. In the *horizontal step*, the check nodes update their outgoing messages  $r_{i\rightarrow j}^0$  and  $r_{i\rightarrow j}^1$ , which corresponds to the probability of check  $i$  being satisfied if  $\hat{y}_j$  is fixed equal to 0 or 1 respectively, while the other adjacent variable nodes in  $\Omega(i)\setminus j$  are subject to the probabilities  $q_{j\rightarrow i}^0$  and  $q_{j\rightarrow i}^1$ . The *vertical step* performs an update of those exact probabilities,  $q_{j\rightarrow i}^0$  and  $q_{j\rightarrow i}^1$ , of variable node

$j$  being either 0 or 1, based on the incoming messages from the check nodes in  $\Phi(j)\setminus i$ .

The whole process is initialized with the received vector  $\hat{\mathbf{y}}$  and the channel statistics, which in the case of a binary symmetric channel is the transition probability  $p$ . From these, the *a priori* probabilities that the transmitted variable  $\hat{\mathbf{x}}_j$  is a 1 or a 0 is calculated.

$$p_j^1 = P(\hat{\mathbf{x}}_j = 1|\hat{\mathbf{y}}_j) = p + \hat{\mathbf{y}}_j \cdot (1 - 2p) \quad (4.16)$$

$$p_j^0 = P(\hat{\mathbf{x}}_j = 0|\hat{\mathbf{y}}_j) = 1 - p_j^1 \quad (4.17)$$

Before the first horizontal step the probabilities  $q_{j \rightarrow i}^x$  are initialized to  $p_j^x \forall i$ .

The algorithm then proceeds with alternating horizontal and vertical steps. As mentioned, the probabilities  $r_{i \rightarrow j}^x$  to be updated in the horizontal steps are the probabilities of check  $i$  being satisfied with  $\hat{\mathbf{y}}_j$  fixed equal to  $x$  and the rest of the adjacent variable nodes in  $\Omega(i)\setminus j$  governed by the probabilities  $q_{j' \rightarrow i}^x$ . The check is satisfied if the calculated parity is equal to zero, as one would expect from a codeword.

The probability  $r_{i \rightarrow j}^x$  then becomes the sum of all the possible configurations of variables values,  $\hat{\mathbf{y}}_{j'}$  for  $j' \in \Omega(i)\setminus j$  that would satisfy check  $i$  while  $\hat{\mathbf{y}}_i$  is fixed, weighted with the inherent probabilities of such configurations. We denote the set of those configurations  $\{\hat{\mathbf{y}}_{j'}|\hat{\mathbf{y}}_j = x\}$ . Hence the following expression for  $r_{i \rightarrow j}^x$ :

$$r_{i \rightarrow j}^x = \sum_{\{\hat{\mathbf{y}}_{j'}|\hat{\mathbf{y}}_j = x\}} \prod_{j' \in \Omega(i)\setminus j} q_{j' \rightarrow i}^{\hat{\mathbf{y}}_{j'}}. \quad (4.18)$$

By evaluating the Fourier transforms of  $r_{i \rightarrow j}^x$ [67] or by a simple induction proof[7], the following simplification of Eq. (4.18) can be deduced:

$$r_{i \rightarrow j}^0 - r_{i \rightarrow j}^1 = \prod_{j' \in \Omega(i)\setminus j} (q_{j' \rightarrow i}^0 - q_{j' \rightarrow i}^1). \quad (4.19)$$

Since  $r_{i \rightarrow j}^0 + r_{i \rightarrow j}^1 = 1$ , Eq. (4.19) is sufficient to find both probabilities in the horizontal step.

In the subsequent vertical step, two procedures are performed more or less simultaneously.

- The variable nodes update their outgoing messages along each edge,  $q_{j \rightarrow i}^x$ , based on the a priori probability and the incoming messages from the check nodes.

$$q_{j \rightarrow i}^x = \alpha_{ij} p_j^x \prod_{\Phi(j)\setminus i} r_{i \rightarrow j}^x \quad (4.20)$$

The constant  $\alpha_{ij}$  is a normalization factor ensuring that  $q_{j \rightarrow i}^0 + q_{j \rightarrow i}^1 = 1$ .

- In addition, each variable node calculates the *pseudo-posterior probabilities*,  $q_j^x$ , which at iteration  $l$  represents the most updated probability that  $\hat{\mathbf{x}}_j = x$  based on the a priori probability and the soft messages from the Tanner-graph out to depth  $2l$ .

$$q_j^x = \alpha_j p_j^x \prod_{\Phi(j)} r_{i \rightarrow j}^x \quad (4.21)$$

With  $\alpha_j$  normalizing the probabilities.

At this point in the algorithm, each variable node makes a hard decision on  $\hat{\mathbf{x}}_j$ ,  $\tilde{\mathbf{x}}_j$ , based on the pseudo-posterior probabilities. If  $q_j^0 > q_j^1$ , then  $\tilde{\mathbf{x}}_j = 0$  and vice versa. This "best guess" on  $\hat{\mathbf{x}}$  is then put to the test by computing the syndrome it would produce. If  $\mathbf{H}\tilde{\mathbf{x}}^T = 0$ , then the algorithm halts having produced a codeword that is assumed to be correct. If the algorithm does not produce a codeword after a predetermined maximum number of iterations, it halts and declares a decoding failure.

As previously mentioned, cycles in the graph will deteriorate the accuracy of the messages. If no cycles are present to depth  $2l$  in the graph, the pseudo-posterior probabilities up to iteration  $l$  will be precise. However, since the end result of this process is the hard decisions on the binary-identities of the variable nodes only, slight inaccuracies in the probabilities are tolerable as long as the nodes decide on the correct bit value.

#### 4.4.2 Log-likelihood ratio

In the binary case it is unnecessary for each node to transmit two messages in each iteration, as  $\text{Prob}(0)$  uniquely determines  $\text{Prob}(1) = 1 - \text{Prob}(0)$ . By redefining the messages as *log-likelihood ratios*, the whole process can be greatly simplified since products in the logarithm-domain become sums. This can lead to significant computational gains.

We start by defining the messages in the LLR-domain, using the natural logarithm as convention:

$$P_j = \log \left( \frac{p_j^0}{p_j^1} \right) \quad R_{i \rightarrow j} = \log \left( \frac{r_{i \rightarrow j}^0}{r_{i \rightarrow j}^1} \right) \quad (4.22)$$

$$Q_{j \rightarrow i} = \log \left( \frac{q_{j \rightarrow i}^0}{q_{j \rightarrow i}^1} \right) \quad Q_j = \log \left( \frac{q_j^0}{q_j^1} \right) \quad (4.23)$$

With these definitions the vertical step in Eq. (4.20) and (4.21) simplifies to mere summations:

$$Q_{j \rightarrow i} = \log \left( \frac{q_{j \rightarrow i}^0}{q_{j \rightarrow i}^1} \right) = \log \left( \frac{\alpha_{ij} p_j^0 \prod_{\Phi(j) \setminus i} r_{i \rightarrow j}^0}{\alpha_{ij} p_j^1 \prod_{\Phi(j) \setminus i} r_{i \rightarrow j}^1} \right) = P_j + \sum_{\Phi(j) \setminus i} R_{i \rightarrow j} \quad (4.24)$$

$$Q_j = \log \left( \frac{q_j^0}{q_j^1} \right) = \log \left( \frac{\alpha_j p_j^0 \prod_{\Phi(j)} r_{i \rightarrow j}^0}{\alpha_j p_j^1 \prod_{\Phi(j)} r_{i \rightarrow j}^1} \right) = P_j + \sum_{\Phi(j)} R_{i \rightarrow j} \quad (4.25)$$

Obviously, the normalization factors are no longer needed as we are only interested in the ratio between the probabilities.

The hard decision on  $\tilde{\mathbf{x}}_j$  is now based on the sign of  $Q_j$ , since if  $q_j^0 > q_j^1$  then  $q_j^0/q_j^1 > 1$ , which leads to  $Q_j > 0$ . If  $q_j^0 < q_j^1$  then  $Q_j < 0$ . The sign gives the hard decision, while the magnitude of  $Q_j$  provides the reliability of said decision.

The horizontal step can also be concisely defined if the following observation is put to use:

$$R_{i \rightarrow j} = \log \left( \frac{r_{i \rightarrow j}^0}{r_{i \rightarrow j}^1} \right) \Rightarrow r_{i \rightarrow j}^0 = \frac{e^{R_{i \rightarrow j}}}{1 + e^{R_{i \rightarrow j}}}, \quad r_{i \rightarrow j}^1 = \frac{1}{1 + e^{R_{i \rightarrow j}}} \quad (4.26)$$

$$r_{i \rightarrow j}^0 - r_{i \rightarrow j}^1 = \frac{e^{R_{i \rightarrow j}} - 1}{e^{R_{i \rightarrow j}} + 1} = \tanh \left( \frac{R_{i \rightarrow j}}{2} \right) \quad (4.27)$$

Since the same relationship holds for  $q_{j \rightarrow i}^x$ , we get the following expression for the update rule of the check nodes when it is applied to Eq. (4.19):

$$\tanh \left( \frac{R_{i \rightarrow j}}{2} \right) = \prod_{j' \in \Omega(i) \setminus j} \tanh \left( \frac{Q_{j \rightarrow i}}{2} \right) \quad (4.28)$$

$$R_{i \rightarrow j} = 2 \cdot \tanh^{-1} \left( \prod_{j' \in \Omega(i) \setminus j} \tanh \left( \frac{Q_{j \rightarrow i}}{2} \right) \right) \quad (4.29)$$

Because of the monotonic and anti-symmetrical features of both the  $\tanh(x)$  and  $\tanh^{-1}(x)$  functions, the sign can be extrapolated giving the most commonly used expression for check node message updates:

$$R_{i \rightarrow j} = \left( \prod_{j' \in \Omega(i) \setminus j} \text{sign}(Q_{j \rightarrow i}) \right) 2 \tanh^{-1} \left( \prod_{j' \in \Omega(i) \setminus j} \tanh \left( \frac{|Q_{j \rightarrow i}|}{2} \right) \right). \quad (4.30)$$

### 4.4.3 Complexity

The main motivation to switch to the LLR-domain is the reduced computational complexity. If we regard a regular code in the probability domain, the computation per iteration scales as follows:

- Horizontal step : Each check node performs  $d_c$  multiplications and then  $d_c$  divisions to calculate the separate messages.
- Vertical step : Each variable node performs  $2d_v$  multiplications to produce  $q_j^0$  and  $q_j^1$ , and the messages  $q_{j \rightarrow i}^0$  and  $q_{j \rightarrow i}^1$  can be determined through  $2d_v$  divisions.

As is clear in the LLR-domain, the computations in the vertical step are negligible while the horizontal step is more complex. The messages  $R_j$  require  $d_c$  hyperbolic tangent evaluations, and  $d_c$  inverse hyperbolic tangent evaluations. This performance can be improved with a *forward-backward algorithm*, to be presented in Section 5.2.1.

Converting to the LLR-domain transfers practically all the complexity of the algorithm to the horizontal step. For that reason, there has been many studies[68] attempting to reduce the computational load of Eq. (4.30). Such attempts will inevitably incur increased probability of decoding failure, as the messages in the decoding algorithm will be degraded, but in some cases the need for efficiency might outweigh the small loss in precision.

One notable contribution is the *Min-Sum Approximation*. In the interval  $x \in [0, \infty)$ ,  $0 \leq \tanh(|x|) < 1$  and  $\tanh(|x|) < |x|$ . Hence, the product of hyperbolic tangents in Eq. (4.30) will be dominated by the smallest term, giving rise to the approximation:

$$\prod_{j' \in \Omega(i) \setminus j} \tanh\left(\frac{|Q_{j \rightarrow i}|}{2}\right) \approx \min_{j' \in \Omega(i) \setminus j} \tanh\left(\frac{|Q_{j \rightarrow i}|}{2}\right), \quad (4.31)$$

which in turn gives the following approximation for the check node update:

$$R_{i \rightarrow j} \approx \left( \prod_{j' \in \Omega(i) \setminus j} \text{sign}(Q_{j \rightarrow i}) \right) \min_{j' \in \Omega(i) \setminus j} |Q_{j \rightarrow i}|. \quad (4.32)$$

Using this approximation the whole algorithm reduces to summations and comparisons. It is apparent that the sign in the messages from the check nodes are correct, however the reliability is reduced. This will lead to greatly reduced complexity per iteration, but might require more iterations in order to recover the codeword.

## 4.5 Density Evolution

According to Shannon's noisy coding theorem[13], there is an upper bound on how low the information rate needs to be in order to reliably decode information sent across a channel with a certain noise ratio. For a BSC( $p$ ), the Shannon limit is  $1 - h(p)$ , as discussed in Section 2. Conversely, for a given information rate there is an upper bound on how much noise any error-correction protocol can reliably decode,  $p_{SL}(r)$ . Hence, a critical parameter in an error-correction protocol with a given rate is the decoding *threshold*,  $p_{DEL}$ , which specifies the practical upper limit on how much noise the protocol can decode. The goal being to attain a threshold  $p_{DEL}$  as close as possible to the Shannon limit  $p_{SL}$ .

Finding theoretical thresholds for single codes are of no real interest, as the threshold can just as easily be found through simulations. Single codes constructed in random or pseudo-random ways may have many low-length cycles or linearly dependent rows, which would lower their true information rate. Finding thresholds for ensembles of codes with similar parameters can however be very useful. Averages can easily be made over big ensembles, and statistical inferences can then be made on individual codes.

Already in 1963, Gallager presented an analytical way of determining the threshold of an ensemble of regular codes. Assuming there are no cycles of length  $2l$  or less, it is possible to derive an expression on the fraction of incorrect messages passed at iteration  $l$ . Incorrect messages in this context signifies messages that would influence the receiver towards an incorrect hard decision. This fraction is a recursive function, dependent on the initial transition probability  $p$ . The threshold is defined so that for every  $p < p_{DEL}$ , the fraction of incorrect messages will go to zero as  $l$  increases, but not for  $p > p_{DEL}$ . He also shows that over a BSC( $p$ ) with  $p < p_{DEL}$ , the bit-error probability decreases exponentially in  $n$ .

Given such results on big ensembles of codes it is possible to make strong assumptions on individual codes, as was shown for the BIAWGNC[67]:

- For a specific code in the ensemble, the probability that the fraction of incorrect messages transmitted at iteration  $l$  differs from the ensemble average, decreases exponentially in  $n$ .
- The ensemble average of the fraction of incorrect messages transmitted at iteration  $l$ , converges towards the cycle-free case exponentially fast in  $n$ .

This means that as  $n$  grows, the probability that a specific code from that ensemble will perform close to optimal goes to one.

Adapting the results of Gallager to the irregular case was first carried out by Luby et. al. as late as in 1997[59]. Their work was done on the BEC with

the corresponding decoding algorithm, introducing the idea that irregular graphs might perform better than regular ones.

Once introduced, this technique was quickly implemented on the BSC[60, 69] and refined for the BEC[61]. The next logical step was to modify this procedure to encompass soft-messages, as used in belief propagation[67]. The technique, suitably called *density evolution*, tracks the probability density functions through the decoding process with the same criterion that the fraction of incorrect messages must converge to zero with the number of iterations.

The expansion to irregular codes meant that the ensembles evaluated include codes of same length and same weight distribution, which are assumed to be cycle-free. The recursive functions determining the fraction of incorrect messages now become dependent on the weight distributions. Hence, the threshold of an ensemble becomes equivalent to the threshold of the weight distribution in question. Based on this, several methods have evolved that optimize the weight distribution in order to maximize the threshold. By placing restrictions on the ensemble, like desired rate and block-length, an optimal weight distribution can be derived using linear programming. Such a protocol called *differential evolution* was already implemented by Luby in 1997[59], but the modern resurrection simply called *iterative linear programming* has shown remarkable results on the BIAWGNC[63].

All of these procedures produce a set of  $\lambda(x)$  and  $\rho(x)$ , which gives an optimal threshold  $p_{DEL}$ . An important contribution in this field performs optimization of  $\{r_i\}$  and  $\{b_i\}$  instead of  $\lambda(x)$  and  $\rho(x)$ [65], which simplifies the actual construction procedure.

## 5 Implementation

Due to the excellent properties of LDPC codes shown in the previous section, it is obviously desirable to try to harness those features in the setting of quantum key distribution. However, a conventional implementation of any linear code in such a volatile setting is non-trivial. Most error-correcting codes are made and optimized for classical communication with error rates  $\ll 1\%$ , while QKD should be able to operate with error rates  $\sim 1 - 10\%$ . Given the classical approach in error-correction that the least amount of errors have happened, which might be slightly naive in QKD, severe restrictions have to be placed on the minimum distance of the code in order to correct a malicious attack by Eve.

Considering the severe loss-rate present in practical QKD setups, due to both low detector efficiency and low attenuation of the laser, it is unfeasible for Alice to encode an original random bit-string into a codeword through Eq.(4.2) and attempt to send it across the quantum channel. A conservative assumption of  $\eta = 0.5$  and  $\nu = 0.5$  would result in only  $\approx \frac{1}{4}$  of the qubits actually reaching Bob, and with approximately half of those measured in non-corresponding bases, only an eighth of the codeword would be shared between Alice and Bob. Consequently, a different approach is required.

### 5.1 Forward Error Correction

One way to overcome this problem is to employ a *Slepian-Wolf* decoding scheme[70], originally invented for classical communication. Here, two pieces of information separately correlated with the source are individually transmitted to a decoder, which is able to jointly use both pieces of information to decode the original information from the source. In the case where one of the two pieces of information can be transmitted lossless, it is called the *Wyner's scheme*. These conditions are ideal for QKD when  $\hat{\mathbf{y}}$  is considered to be one of the pieces of information available to the decoder, Bob. The other piece would be the parity information,  $f(\hat{\mathbf{x}})$ , exchanged in post-processing across the public channel. Hence, the error-correction part of a QKD scheme can be considered a Wyner's scheme.

Using the CASCADE-protocol described in Section 3.1.4, the redundant information  $f(\hat{\mathbf{x}})$  becomes the information on the parity of the different blocks of  $\hat{\mathbf{y}}$  and  $\hat{\mathbf{x}}$ , and the decoding process is just the corresponding flipping of bits. However, this approximation is not entirely accurate since the Wyner's scheme in principle describes one-way communication or *forward error-correction (FEC)*.

LDPC codes on the other hand are very suitable to be used in this setting[71]. In classical coding theory a message is encoded into a codeword,  $\hat{\mathbf{c}}$ , such that  $\mathbf{H} \cdot \hat{\mathbf{c}}^T = \hat{\mathbf{0}}$ . In order to use LDPC codes for Slepian-Wolf decoding, one conceptual modification needs to be done. First it is important to realize that the subspace

of the  $n$ -dimensional space defined by the code so that  $\mathbf{H} \cdot \hat{\mathbf{x}}^T = \hat{\mathbf{0}}$ , is arbitrary and any subspace defined by  $\mathbf{H} \cdot \hat{\mathbf{x}}^T = \hat{\mathbf{s}}$  can be used in the exact same way. The different subspaces defined by different syndromes,  $\hat{\mathbf{s}}$ , are called *cosets*. Decoding in the normal way can be regarded as *decoding within the  $\hat{\mathbf{0}}$ -coset*.

The conceptual modification mentioned above, is not restricting the protocol to do decoding only within the  $\hat{\mathbf{0}}$ -coset but within the  $\hat{\mathbf{s}}$ -coset, where  $\hat{\mathbf{s}}$  is given by  $\mathbf{H} \cdot \hat{\mathbf{x}}$ . In that way, the received vector  $\hat{\mathbf{y}}$  can be regarded as a faulty version of a "codeword",  $\hat{\mathbf{x}}$ , within the  $\hat{\mathbf{s}}$ -coset. For Bob to be able to decode  $\hat{\mathbf{y}}$ , he obviously needs to know which coset he should decode in. Consequently, the redundant information given on  $\hat{\mathbf{x}}$ ,  $f(\hat{\mathbf{x}})$ , is the syndrome-vector,  $\hat{\mathbf{s}} = \mathbf{H} \cdot \hat{\mathbf{x}}^T$ , defining the coset in which  $\hat{\mathbf{x}}$  can be regarded as a codeword.

In comparison to the CASCADE-protocol,  $\hat{\mathbf{s}}$  is similarly a collection of parity-information on  $\hat{\mathbf{x}}$  of the blocks defined by the rows of  $\mathbf{H}$ . However, when using the framework of linear codes it is possible to take advantage of the optimal decoding strategies and the promise of performance extremely close to the Shannon limit[10].

## 5.2 Programming

From a computational standpoint, the main focus is keeping the protocols as time-efficient as possible. Consequently, the logical choice of programming language is C++ because it allows object-oriented programming while being simple, robust and computationally efficient. Both the PEG-algorithm of Section 4.3.3 and the LLR-based BP-algorithm of Section 4.23 has been implemented in relation to this thesis, and expressing the check- and variable nodes as objects dramatically simplifies the structure of both these algorithms. The structure of the check node-object can be seen in Listing 1 in Appendix A.

### 5.2.1 Decoding algorithm

The main challenge in the implementation of the decoding algorithm is the horizontal step, summarized in Eq. (4.30). The multiple  $\tanh$ - and  $\tanh^{-1}$ -evaluation needed in order to calculate the outgoing message from the check nodes are computationally costly and constitutes the main bulk of the computational load. Thus, it is of great interest to reduce the amount of such calculations as much as possible. A simple way of doing this is to use a *forward-backward-pass*-approach, which is implemented in Listing 2. If check node  $i$  has the incoming messages,  $Q_{j \rightarrow i}$ , through  $d_c$  edges, the vectors  $\hat{\mathbf{F}}$  and  $\hat{\mathbf{B}}$  can be defined in the following way:

$$\hat{\mathbf{F}}_j = \hat{\mathbf{F}}_{j-1} \cdot \tanh^{-1} \left( \frac{Q_{j \rightarrow i}}{2} \right) = \prod_{k=1}^{k=j} \tanh^{-1} \left( \frac{Q_{j \rightarrow i}}{2} \right) \quad (5.1)$$

$$\hat{\mathbf{B}}_j = \hat{\mathbf{B}}_{j+1} \cdot \tanh^{-1} \left( \frac{Q_{j \rightarrow i}}{2} \right) = \prod_{k=j}^{k=d_c} \tanh^{-1} \left( \frac{Q_{j \rightarrow i}}{2} \right), \quad (5.2)$$

where  $\hat{\mathbf{F}}_0 = \hat{\mathbf{B}}_{d_c+1} = 1$ . This allows for a very compact expression for  $R_{i \rightarrow j}$ .

$$R_{i \rightarrow j} = 2 \cdot \tanh^{-1} \left( \hat{\mathbf{F}}_j \cdot \hat{\mathbf{B}}_{j+1} \right) \quad (5.3)$$

In contrast to a brute force approach to Eq. (4.30), which would yield  $d_c \cdot (d_c - 1)$  inverse tanh evaluations, the forward-backward approach gives only  $2d_c$  evaluations. One of the codes implemented in this text has an average check node weight over 50, so with 10.000 check nodes and over 20 iterations, the decoding algorithm is spared of 470 million inverse tanh evaluations in each pass of the BB84-protocol. The remaining evaluations can be done efficiently using a look-up table, as suggested by Chen et. al[72].

The inverse hyperbolic tangent function has the following behavior:  $\lim_{x \rightarrow \pm 1} \tanh^{-1}(x) = \pm \infty$ . Thus, in order to keep the messages finite, it is commonplace to implement *cutting*, making sure the input to the inverse hyperbolic tangent does not exceed a set limit. Typically this limit is set very close to  $\pm 1$ , e.g.  $\pm(1 - 10^{-12})$ .

The belief propagation decoding algorithm promises accurate probabilities at any given time, assuming cycle-free graphs. In the transition to practical implementations however, the messages are subject to the finite precision of computers. This could in theory degrade the performance of the algorithm. For this reason, the implementations in this text has used a 53 bit representation of each message and probability, which should be more than enough to avoid severe round-off errors[73].

In order to apply the BP-algorithm in Wyner's scheme as described above, one crucial modification needs to be done. Since the BP-algorithm is constructed in the framework of classical coding-theory, it automatically assumes  $\hat{\mathbf{0}}$ -syndrome decoding. This is expressed in the outgoing message  $R_{i \rightarrow j}$  from Eq. (4.23), where the probability of the check node being satisfied is equated to the probability of it being zero. If the true syndrome value of node  $i$ ,  $\hat{\mathbf{s}}_i = 1$ , then the expression of  $R_{i \rightarrow j}$  must be flipped, leading to the following modification:

$$R_{i \rightarrow j} = (-1)^{\hat{\mathbf{s}}_i} \log \left( \frac{r_{i \rightarrow j}^0}{r_{i \rightarrow j}^1} \right). \quad (5.4)$$

### 5.2.2 Codes

To truly test the performance of LDPC codes as an error-correction scheme for QKD it was imperative to have good codes. As previously discussed, it is important to have a good edge placement protocol in order to avoid cycles, as well as optimized irregular degree distributions to ensure the best possible performance. The PEG-algorithm implemented as part of this work handled the first of these issues. It is easily adaptable to length, rate and degree distribution, in addition to supporting both greedy and non-greedy construction.

Due to the recent boost of interest in LDPC codes, many optimized degree distributions are publicly available, both in articles[63, 74] and online[75, 76]. The latter two webpages also offers both individually specified density evolution and differential evolution for several different channels. The degree distributions used in this text for the different rates can be found in Appendix A along with their respective sources.

### 5.2.3 Simulations

To test the fully constructed codes, they were subjected to comprehensive simulations. A pseudo-random number generator was used to create a more or less random  $n$ -bit string. After the syndrome for this string was computed, a certain fraction of errors was introduced using the same pseudo-random number generator. The faulty string was passed to the decoding algorithm along with the true syndrome and the fraction of errors. Starting out at 0, the fraction of errors was increased in 0.1% intervals until it reached the shannon limit  $p_{SL}$  for the given information rate of the code. For each error-fraction this process was repeated for 1000 iterations or until 100 decoding errors was observed, every time with a new distribution of errors. The max limit on the number of iterations for the decoding algorithm was set at 1000.

## 5.3 Results

In accordance with the weight distributions in Appendix A, 12 different codes were generated using the non-greedy PEG-algorithm. Every code has equal length,  $n = 100.000$  and equal girth,  $g = 6$ . Using the simulation technique described above, a simple connection was found between the fraction of corrected blocks as a function of the introduced fraction of bit errors. The introduced QBER obviously corresponds directly to the transition probability,  $p$ , of a BSC( $p$ ). The plot in Fig. 5.1 shows one of these connections for the code of rate  $r = 0.5$ .

At this point, it is important to emphasize that the fraction of corrected strings for all codes and for all error-rates below their respective *observed limits* were

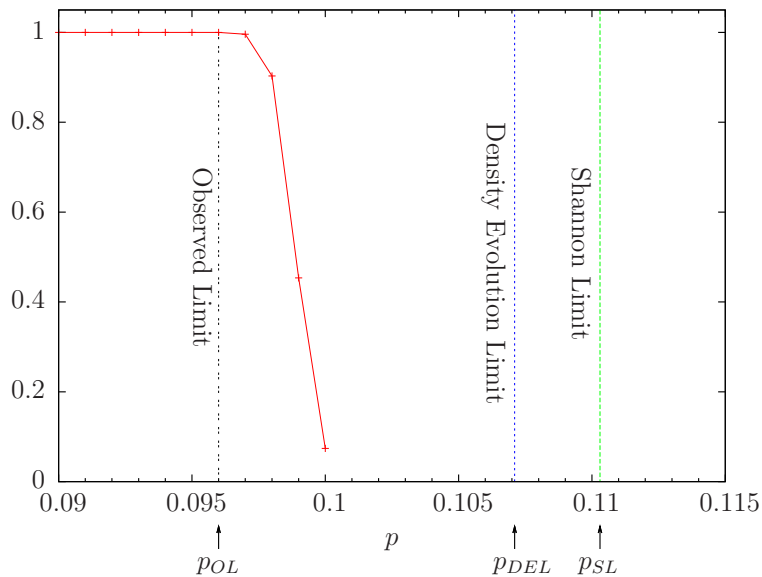


Figure 5.1: The corrected fraction of a code with  $r=0.5$ ,  $n=100.000$ ,  $g=6$  and  $\vec{r}, \vec{b}$  according to table 3 in Appendix A, plotted as a function of the transition probability  $p$ . Also plotted is the limit according to density evolution for this degree distribution and the Shannon limit for  $r=0.5$  as well as the observed limit.

without exception identically equal to one. In fact, the definition of the observed limit,  $p_{OL}$ , for a specific code is the highest transition probability,  $p$ , for which the code corrected 100% of the 1000 simulations. The Shannon limit,  $p_{SL}$ , for a code with rate  $r$  is defined as  $C(p_{SL}) = r$ , with  $C(p)$  from Eq. (2.5), while the limits found from density evolution,  $p_{DEL}$ , are those provided from the source of the degree distribution.

Another important aspect is that in these regions below or at the observed limit, no incidents of undetected errors were made for any code.

An apparent trend in Fig. 5.1 is that the code exhibits only a slight reduction in the corrected fraction in the  $p$ -range directly following the observed limit. This is a feature observed in many of the codes. The amount of remaining errors in the uncorrected blocks after 1000 iterations were highly diverse, but contrary to other findings[74], no occurrences of very few remaining errors ( $< 10\%$ ) was observed.

All the codes with their respective limits are summarized in Fig. 5.2 and compared to the Shannon limit and the performance of CASCADE[9, 31]. Each horizontal line represents an individual code that can correct up until its observed limit. For higher values of  $p$ , a code with lower rate has to be used, resulting in a step function.

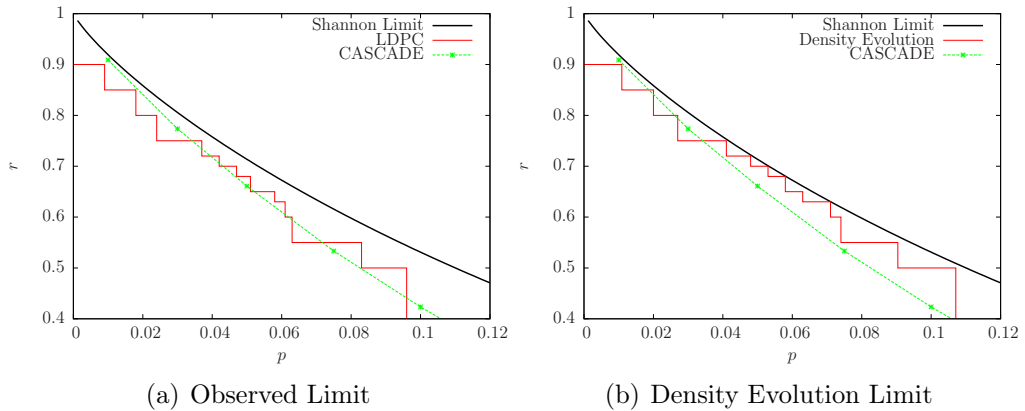


Figure 5.2: Information rate as a function of  $p$  using LDPC codes and CASCADE.

A different perspective that proves descriptive, is portraying the same information using the efficiency parameter,  $\kappa(p)$ , introduced in Section 3.1.5. The observed values from these simulations are shown in Fig. 5.3.

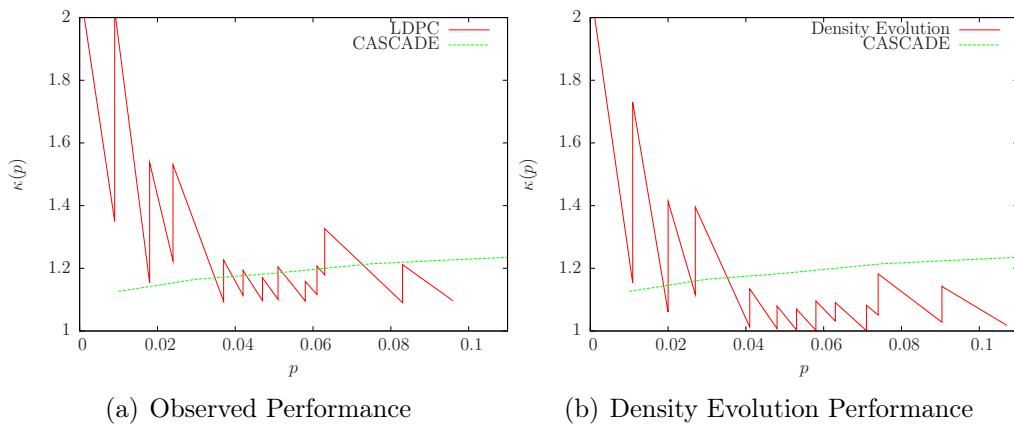


Figure 5.3:  $\kappa(p)$  as a function of  $p$ , calculated from both the observed performance and the density evolution performance.

The plots in Fig. 5.2(a) and Fig. 5.3(a) show the true performance of the implemented error-correction protocol. Fig. 5.2(b) and Fig. 5.3(b) are added for illustrative purposes and show the corresponding hypothetic plots that would have occurred if the codes performed as well as their density evolution limits propose.

During the simulations, the number of iterations needed and the time spent to decode, was averaged over all 1000 successful attempts. This was done for all levels of QBER, and for all codes as can be seen in Fig. 5.4.

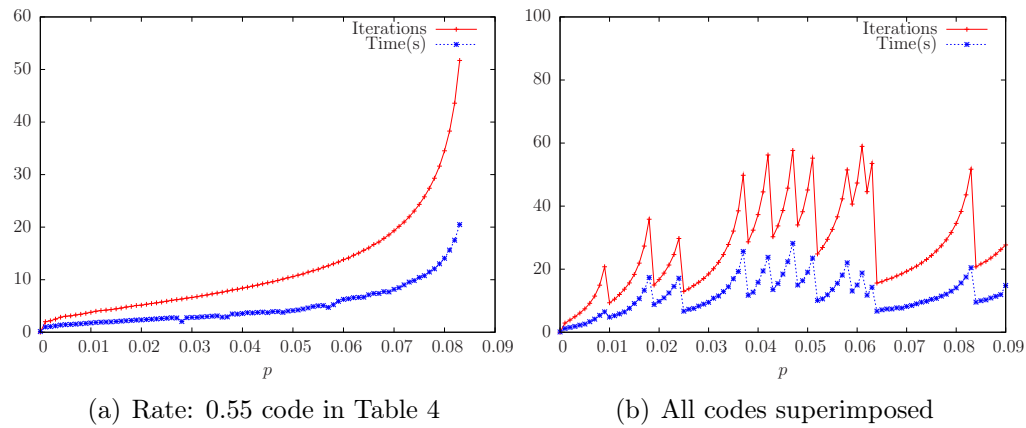


Figure 5.4: The average number of iterations needed, and the average time taken (in seconds) in order to decode successfully.

The plot in Fig. 5.4(a) shows how the average time clearly depends on the number of iterations, and how they both increase dramatically close to the Shannon limit for this code. The same trend is apparent for all codes, which can be seen in Fig. 5.4(b). In this plot the time and iteration data at a certain QBER,  $\tilde{p}$ , comes from the code with highest rate, which achieved 100% correct decoding at  $p = \tilde{p}$ .



## 6 Discussion

The most important characteristics of an error-correction protocol from a theoretical point of view, are how close it can perform to the Shannon limit, the computational cost and its decoding speed. From a practical point of view it is also important to review how easily it can be implemented and run. After Brassard and Salvail[9] proposed their CASCADE-protocol as a simple, low complexity alternative to the computationally heavy classical error-correcting codes at the time, it has become the de facto choice for most QKD implementations and still is. Consequently, it is the obvious point of reference to the performance of the LDPC protocol implemented in this text.

Unfortunately, there is little literature available directly comparing these two protocols or comprehensively examining the CASCADE-protocol. Due to scarcity of time it was not possible to implement it for the purpose of this text, hence the following evaluation will be based on the few reference points that are available.

### 6.1 Kappa

When using LDPC codes as the error-correction protocol in QKD, the rate,  $r$ , of the specific code used, decides the amount of redundant parity information sent,  $|\hat{s}| = n(1 - r)$ . Clearly, it is desirable to use the highest rate possible. In practice, this works by Alice and Bob estimating the QBER of the block,  $p^*$ , as described in Section 3.1.2. This estimate determines the rate of the code to be used.

Earlier implementations of LDPC-codes with QKD[11, 12, 77] have used a system where Alice and Bob generate a new code for each key reconciliation, dependent on the block length and the observed QBER. In order to make this system efficient, these codes has to be constructed as quickly as possible. Hence, the advanced algorithms to generate optimal codes are unfeasible, leaving only a pseudo-random construction scheme where Alice and Bob agree on a seed for a pseudo-random number generator and use a regular degree distribution. As described in Section 4, such codes are unable to perform arbitrarily close to the Shannon limit. In fact, the density evolution limit for the often used  $(n,3,6)$ -regular code with  $r = 0.5$ , is  $p_{DEL} = 0.088$ [67]. Consequently, Elliott et al.[11] describe using an extra margin on the rate for suboptimal codes, in the construction phase. Their reports are thus of higher values of  $\kappa$  for LDPC codes compared to their implementation of CASCADE, exemplified by  $\kappa(0.03) = 1.25$  for their LDPC-protocol and  $\kappa(0.03) = 1.20$  for their CASCADE-protocol. Lodewyck et al.[12] implemented LDPC codes with a continuous variable QKD-protocol and report performance of  $\kappa(p) > 1.10$ .

In 2009, Matsumoto et al.[78] presented a paper outlining the weaknesses of using LDPC codes for QKD. One flaw being that Alice and Bob does not have

optimized codes available to them at the time of decoding. The conceptual difference between the earlier protocols and the one presented in this paper, is utilizing a set of pre-established codes, an idea proposed by Elkouss[74].

The plots in Fig. 5.2 and Fig. 5.3 show the efficiency of the LDPC codes used in this text, compared to the CASCADE-protocol[9]. An apparent trend is the poor performance of the LDPC codes for  $p < 3\%$ , which is exactly the region where CASCADE performs best. In fact,  $\kappa(0.03) = 1.286$  for this protocol. The reason for the high values of kappa in this region is the definition:  $\kappa(p) = (1 - r)/h(p)$ . Thus, LDPC codes in low QBER-regions needs to perform closer to their density evolution limits in order to give the same values for  $\kappa$ .

Once the QBER reaches 3.7%, the value of  $\kappa$  is reduced to only 1.095. After this limit, there is a region of QBER ( $0.037 < p < 0.063$ ) where the LDPC protocol performs better or marginally worse than CASCADE, with the excess of parity information spanning from 9.5% to 22.6% above the Shannon limit, which is evident in Fig. 5.3(a). This region of good performance coincides superbly with the level of QBER, 3-6%, which is normally experienced in QKD setups[79].

The reason why the performance is so good in this region is due to the high density of codes with observed limits evenly distributed in this exact QBER-region, resulting in a higher resolution step-function. The small peak in  $\kappa$  at  $p \approx 0.063$  could be avoided in the same way by introducing more codes with  $r \in [0.5, 0.6]$ , which should approximately correspond to  $p_{OL} \in (0.063, 0.110)$ . This would make the protocol superior to CASCADE with regards to  $\kappa(p)$  for all  $p > 0.037$ . The same approach would obviously also work for  $p < 0.037$ , only not to the same extent due to the definition of  $\kappa$ . It could reduce the peaks in Fig. 5.3(a), but not lower the local minima.

In stead of increasing the frequency of the steps in Fig. 5.2(a), the other apparent way to improve this protocol is to increase the protrusion of each step. Fig. 5.2(b) shows the same step function with the same codes, only using the limit imposed by density evolution as opposed to the observed limit. These limits,  $p_{DE}$ , are tight upper bounds on the achievable observed limit,  $p_{OL}$ , of codes constructed using the corresponding degree distribution. These limits are only achievable in the asymptotic limit of infinite block length and with cycle-free graphs. Consequently, it is hard to predict the actual performance of such codes in realistic settings, as well as predicting the increase in block length and girth needed in order to approach the density evolution limits. It is in this setting that the true strength of this protocol is revealed. Since the codes are constructed beforehand, Alice and Bob has in principle infinite time to construct and refine their codes, and even obtain better degree distributions. This gives such an LDPC-based protocol the possibility of performing arbitrarily close to the Shannon-limit, which will be discussed further in Section 6.5.

## 6.2 Speed

In the cases where LDPC codes have been preferred to CASCADE despite lower rates[11, 12, 77, 80, 81], the main argument has been the decoding speed. Due to the highly interactive nature of both CASCADE and its modifications[28, 29, 31], it can become a bottleneck if the experiment is looking to achieve high key rates.

According to the description of the belief propagation algorithm of Section 4.4.1, the computational complexity of each node is independent of the block size and the QBER. Consequently, only three factors dictate the speed of convergence for the BP-algorithm:

- The computational complexity in each node, both for the check and variable nodes.
- The total amount of nodes, which is fixed for a specific code.
- The number of iterations needed for the decoding algorithm to converge.

In these simulations, the first two parameters are fixed, which only leaves the number of iterations dictating the time spent. This connection is clearly depicted in both Fig. 5.4(a) and Fig. 5.4(b). The average decoding time over all QBER from the data in Fig. 5.4(b), is  $< 20$  s for  $n = 100.000$ , which means that this protocol can perform decoding with a key-rate in excess of 5kbit/s.

The plot in Fig. 5.4(a) shows an explosive time increase when the QBER approaches the limit of the code. An obvious way to reduce the time and consequently increase the reconciled key rate, is to use the codes further away from their respective limits. This would mean shifting the entire step function of Fig. 5.2 to the left, reducing the information rate and increasing  $\kappa(p)$ . Even though this would result in fewer reconciled key-bits per pass of the BB84, the lowered time could make up for it in regards to the reconciled key rate. This is under the assumption that all other parts of the entire system is capable of operating at such high speeds, leaving the error-correction as the bottleneck.

Another way of reducing the time is to use an approximative decoding algorithm, like the *min-sum* approach introduced in Section 4.4.3 or other numerical approximations[82]. Such approximations will greatly reduce the computational complexity in each node, at the expense of more iterations and increased failure probability. One easily implementable approximation that could increase the speed of this protocol, is to use look-up tables for the hyperbolic tangent evaluations as described in Section 5.2.1. This has not been done at present, and is expected to increase the speed.

When CASCADE was first introduced, it was an alternative to the heavy calculations of the error-correcting codes at the time, which is still one of the

main arguments for choosing CASCADE over LDPC codes. While the exact BP-algorithm is still computationally expensive, today's computers have made them increasingly feasible. A major advantage is that the BP-algorithm is perfectly suited for parallel computing. In this implementation, the message updates are performed internally in the node-objects and should be easily parallelizable. On an 8-core computer, this could lead to realistic key-rates close to 40 kbit/s, without the added failure probability or increased rates. In fact, LDPC codes have been implemented with high-speed protocols operating at 50 kbit/s[81].

Since CASCADE was not implemented, there is no way of directly comparing the decoding time of these two protocols. However, the speed of CASCADE is fundamentally limited due to the interactivity, while LDPC codes are just limited by the computational power of the computer. It is then reasonable to assume that LDPC codes are not just faster at present, they also have a higher potential for improvement.

### 6.3 Security

Given the security proofs decoupling error-correction from privacy amplification[45, 48], it is possible to evaluate the strengths and weaknesses of these protocols separately. This also entails the security perspective.

By encrypting the parity information transmitted during reconciliation as suggest in Section 3.1.5, this process will not directly reveal any information on the key to Eve. However, when using a set of pre-defined codes it has to be assumed that Eve has full knowledge of these codes as well as the QBER-intervals in which they are used. This could possibly open a security loop-hole for Eve. While she can always introduce enough errors to break the protocol completely and cause a detected error, the real danger in this scenario is if she can manipulate the key in such a way that an undetected error occurs. That would mean that Alice and Bob believe they have identical keys, when in fact they do not, which could have disastrous effects for the entire protocol.

If Eve knows the code that is used, it is conceivable that she could target single bits in hopes of incurring an undetected error. The PEG-algorithm described in Section 4.3.3 is constructed in a way so that the last variable nodes are the ones with the highest degrees, consequently being the most influential in the decoding algorithm, affecting many check nodes. An attack specifically targeting those bits could cause the decoding algorithm to converge against an incorrect key. Such a specific attack is practically infeasible in the face of severe deficiencies in the components of the setup, which makes it close to impossible to know which photon will end up as which bit in the sifted key. However, allowing Eve full control over the quantum channel requires very conservative assumptions on Eve's abilities. For that reason, Alice and Bob perform random uniformization of the key, proposed in

Section 3.1.2, which ensures that Eve is unable to target specific bits in the sifted key.

In a practical setup it is not desirable to do parameter estimation more than necessary. When a protocol like the BB84 is run over longer periods of time, the QBER usually stays more or less stable. It is therefore sufficient to do estimation with fixed intervals on the BB84-passes, or in the event of a detected error. This opens the possibility for Eve to attack in the periods that are not subject to estimation. If Eve's attack increases the QBER for these blocks beyond the observed limit for the applied code, the most likely outcome is that the reconciliation attempt will fail, the block will be discarded and the parameters are estimated again. Nevertheless, it is possible that Eve has introduced so much noise that  $d(\tilde{\mathbf{x}}, \hat{\mathbf{y}}) < d(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ , for an  $\tilde{\mathbf{x}} \neq \hat{\mathbf{x}}$  s.t.  $\mathbf{H}\tilde{\mathbf{x}}^T = \hat{\mathbf{s}}$ , making the decoding algorithm converge against  $\tilde{\mathbf{x}}$  rather than the true key,  $\hat{\mathbf{x}}$ , causing an undetected error. Eve will not gain any knowledge on  $\hat{\mathbf{x}}$  in this case, but an undetected error is still critical.

This problem can be solved by introducing another step in the protocol called *verification*. After a successful key reconciliation, Alice and Bob can make sure their two keys indeed match up by using the same approach employed for parameter estimation. They choose a small random subset of bits and publicly announce them. If a discrepancy is observed it means that an undetected error has occurred, they discard the block and proceed to do parameter estimation, which would detect Eve. Compared to ordinary estimation, verification requires a much lower fraction to be revealed because with verification, the pair only needs to observe one error in order to identify an undetected error. Additionally, it is possible to exploit the minimum distance of the LDPC codes to further minimize the number of bits to be estimated.

As described in Section 4.1, the Hamming distance between two codewords in a linear code,  $C$ , is greater than, or equal to the minimum distance of that code. That means that if a decoding algorithm reports a successful decoding but has produced an incorrect codeword, the distance between the two can be assumed to be at least  $d_{\min}$ . On the other hand, the strings  $\hat{\mathbf{x}}$  and  $\tilde{\mathbf{x}}$  are not generally codewords of  $\mathbf{H}$ , in the sense that  $\mathbf{H}\hat{\mathbf{x}}^T = \hat{\mathbf{s}}$  is not generally equal to  $\hat{\mathbf{0}}$ . The following proof will show why the same argument can be made for  $\hat{\mathbf{x}}$  and  $\tilde{\mathbf{x}}$ .

If the BP-algorithm is assumed to produce an undetected error as described above, then  $\mathbf{H}\hat{\mathbf{x}}^T = \mathbf{H}\tilde{\mathbf{x}}^T = \hat{\mathbf{s}}$ , with  $\hat{\mathbf{x}} \neq \tilde{\mathbf{x}}$ . This means that:

$$\mathbf{H}(\hat{\mathbf{x}} \oplus \tilde{\mathbf{x}})^T = \hat{\mathbf{s}} \oplus \hat{\mathbf{s}} = \hat{\mathbf{0}}, \quad (6.1)$$

meaning that  $(\hat{\mathbf{x}} \oplus \tilde{\mathbf{x}}) \in C$ . Consequently, the distance can be given in the following way:

$$d(\hat{\mathbf{x}}, \tilde{\mathbf{x}}) = \text{wt}(\hat{\mathbf{x}} \oplus \tilde{\mathbf{x}}) \geq d_{\min} \quad (6.2)$$

Since  $(\hat{\mathbf{x}} \oplus \tilde{\mathbf{x}})$  can be regarded as a codeword in  $C$ , its weight must follow the restriction of Eq. (4.6). Any code,  $C$ , defined by a matrix,  $\mathbf{H}$ , so that  $\{\hat{\mathbf{c}} \in C | \mathbf{H}\hat{\mathbf{c}}^T = \hat{\mathbf{0}}\}$  is always linear, since  $\mathbf{H}(\hat{\mathbf{c}} \oplus \tilde{\mathbf{c}})^T = \hat{\mathbf{0}}$  for  $(\hat{\mathbf{c}} \neq \tilde{\mathbf{c}}) \in C$  and  $\mathbf{H}(\beta\hat{\mathbf{c}})^T = \hat{\mathbf{0}}$  for any constant  $\beta$ , fulfilling the linearity restriction. Consequently, the prior result holds for any LDPC code.

In the event of an undetected error, the above result states that the two vectors are at least different in  $d_{\min}$  bits. If they sample a fraction,  $\gamma$ , of the keys, the probability,  $P_{II}$ , that an undetected error stays undetected after verification can be described by a simple hypergeometric distribution:

$$P_{II} \leq \frac{\binom{n-d_{\min}}{\gamma n} \cdot \binom{d_{\min}}{0}}{\binom{n}{\gamma n}} \quad (6.3)$$

$$= \frac{\frac{(n-d_{\min})^{\gamma n}}{(\gamma n)!}}{\frac{n^{\gamma n}}{(\gamma n)!}} = \left(1 - \frac{d_{\min}}{n}\right)^{\gamma n} \quad (6.4)$$

The obstacle is now to find the minimum distance of a code. As was described in Section 4.2, this problem is exponentially hard, exemplified by Eq. (4.6) which would require  $2^m$  evaluations to be done. It is however possible to utilize the known characteristics of classical codes from Section 4.1. A linear code can not correct more errors than  $\lfloor \frac{d_{\min}}{2} \rfloor$ , while these codes have been shown to correct  $p_{OL} \cdot n$  errors, which means:

$$p_{OL} \cdot n \leq \lfloor \frac{d_{\min} - 1}{2} \rfloor \leq \frac{d_{\min} - 1}{2} \quad (6.5)$$

$$\Downarrow \quad (6.6)$$

$$d_{\min} \geq 2 \cdot p_{OL} \cdot n + 1. \quad (6.7)$$

Combined with Eq. (6.4), this gives the following expression for  $P_{II}$ :

$$P_{II} \leq \left(1 - 2 \cdot p_{OL} + \frac{1}{n}\right)^{\gamma n} \quad (6.8)$$

Hence, the probability of an undetected error can be made arbitrarily small by increasing the sampling,  $\gamma$ , or in the asymptotic bound for  $n$ .

If LDPC codes and error verification is employed in such a manner, an attack by Eve can be detected and foiled with probability arbitrarily close to unity. However, she can launch a more sinister attack that can not be detected by the scheme described above. When fewer codes are used in an FEC-protocol, every code needs

to cover a bigger span of QBER, e.g. the code of rate 0.55 in this implementation covers the span  $0.064 < p < 0.083$ . In theory, Eve can manipulate the QBER so that Alice and Bob estimates  $p^* = 0.064$ , forcing them to use the rate 0.55 code. If she at that point performs an attack resulting in QBER closer to, but not exceeding the observed limit of the code, the error-correction will succeed and the attack will go unnoticed. Her mutual information with the key, which needs to be removed in privacy amplification is then  $\approx h(p_{OL}) > h(p^*)$ . If privacy amplification is performed using the estimate  $p^*$ , the key is not secure.

This advantage can be removed if Bob calculates the *true* QBER,  $d(\hat{\mathbf{x}}, \hat{\mathbf{y}})/n$ , after a successful reconciliation and verification, which is then shared with Alice and used for privacy amplification. If it exceeds the limit of 11% they discard the block, and in the next pass Alice and Bob can use the true QBER of the previous pass to decide the appropriate code, leaving the estimation protocol almost useless. In the event of a detected error, Alice and Bob could estimate the QBER anew, in order to choose a suitable code. An alternate approach for the next pass could be to simply choose the code with the highest observed limit. It will guarantee a successful decoding for acceptable levels of QBER, followed by a calculation of the true QBER used for the subsequent pass. The same procedure can also be used for the initial pass with a completely unknown QBER.

Such a scheme renders the need for an estimation protocol completely useless. The same approach is also used in CASCADE, where the true number of erroneous bits is known to Bob after a successful reconciliation. Consequently, fewer bits needs to be expended for estimation, at the possible expense of run-time.

The expression for the secure key rate in Eq. (3.9) is one of the most important parameters in QKD. In this equation the privacy amplification protocol was assumed to be perfect, however this is not the case in actual realizations. In most security proofs concerning real-world imperfections[42, 83, 84, 85], the security lies in quantifying the extra amount of information leaked to Eve due to the imperfections, and increase the amount of privacy amplification correspondingly. Hence, the lower  $\kappa(p)$  is, the more privacy amplification can be done without getting a negative secure key rate. In this perspective, the performance of the error-correction protocol is directly related to the security of the protocol.

## 6.4 Disadvantages

Even though LDPC-codes have been shown to perform equally good, or better than CASCADE in most ways, there are certain disadvantages with the scheme proposed in this text.

When implemented in the way described above, this scheme is restricted to a given block-size,  $n$ . This is not a major disadvantage in itself, since it is fairly straightforward to implement a buffer in either the software on the computer or

in the electronics regulating the preparation/detection, so that an  $n$ -sized block of sifted bits is given to the post-processing protocol once the buffer reaches that amount. The main downside of this setup is that it is less flexible than the CASCADE protocol. If the experimentalists want to change the block-size for any reason, they have to first produce a set of codes with the desired size, which can be a tedious task. On the other hand, once the codes are constructed they can be stored and will always be available. That means the longer this system is operated, the more flexible it becomes. When it is desirable to simply test the performance with another block-size, the performance of the system might be well-known. In such cases, it should be sufficient to create only a few codes that can operate around these QBER-levels.

One of the criticisms presented by Matsumoto[78], is that the quantum channel might not always perform like a perfect BSC;  $P(\hat{y} = 1|\hat{x} = 0) \neq P(\hat{y} = 0|\hat{x} = 1)$ . Since the density evolution algorithm depends on these probabilities, the degree distributions found for a BSC might not work optimally in such cases. While it is easy to adapt the decoding algorithm to take such a discrepancy into account, the effect on the overall performance is hard to predict. If the channel is not symmetric but with stable transition probabilities, it is possible to implement a density evolution algorithm specifically tailored for these probabilities.

As mentioned in Section 5.3, there were no observed incidents of the decoding algorithm halting with only a few erroneous bits left. Such a phenomenon has been reported by others[12, 74], who also report on solving the issue quite simply by using an additional classical error-correcting code with very high rate;  $r \approx 0.998$ . This was reported to correct all the remaining errors with very high probability[12, 74]. Since these fallacies were not observed in this text, the additional codes have been deemed redundant. Implementation of such codes would not affect the security conclusions described above as long as the final key has the correct syndrome, but a small reduction in the secret key rate is thus incurred.

## 6.5 Outlook

The ease of implementation has often been a deciding factor in favor of CASCADE compared to LDPC codes. It is conceptually simpler, and needs little maintenance once realized. A protocol employing LDPC codes on the other hand, can be more complex and more time consuming to implement. A protocol like the one suggested here is particularly complex since it requires more sophisticated algorithms for constructing the codes, in contrast to the random construction methods common in previous implementations. However, once the implementation is done, it is fairly straight-forward to construct a set of codes that can outperform CASCADE, as demonstrated in this text. The PEG-algorithm used here was relatively slow, restricting the block-size to 100.000 and the girth to 6. While the codes already

constructed can perform reasonably well and outperform CASCADE in certain QBER-regions, the good prospects provided by density evolution gives reason to believe that the protocol can be drastically improved.

Another important aspect is that LDPC codes is a thriving area of research within the field of classical coding theory, where new ideas in many cases can be directly applicable to this protocol. To the knowledge of the author, little, if any, further development is performed on the CASCADE-protocol.

Further improvements that should be performed on the present software prior to practical use, include parallelizing the software as well as introducing look-up tables for the hyperbolic tangent evaluations. While the general refinement of supplementing and revising the set of codes should be a continuous process, an immediate improvement can be made by constructing two more codes with rates between 0.75 and 0.80, and 0.55 and 0.60 respectively, which should greatly reduce the peaks in  $\kappa$  around 3% and 6.5%.

Since the simulations performed in this text gave no undetected errors even for error-rates above the observed limit, it is reasonable to believe that the true minimum distance of these codes are even higher than the deductions in Section 6.3 suggests. Hence, stronger limits on the minimum distance from theoretical bounds or numerical simulations will provide an even tighter bound on Eq. (6.8), which leads to a higher effective key rate.



## 7 Concluding remarks

In the duration of this text I have given a thorough introduction to the relevant theory of both quantum key distribution and low-density parity-check codes. I have utilized this theory and implemented a fully operational key reconciliation protocol, which breaks with the traditional approach. Through extensive simulations, it has shown very promising performance, even outperforming conventional protocols such as CASCADE for a wide range of parameters.

By employing the theoretical properties of these codes, I have developed a novel detection scheme that in principle makes the action of parameter estimation obsolete. It is more efficient, consuming fewer secret bits, and more adept at detecting Eve since it employs the true QBER and not an estimate.

In light of the observed performance as well as the intense research carried out within the field of classical coding-theory, I am convinced that LDPC codes has a much brighter future than CASCADE as an alternative for key reconciliation in QKD.



## References

- [1] B. Schumacher, “Quantum coding,” *Phys. Rev. A*, Jan 1995.
- [2] W. K. Wootters and W. H. Zurek, “A single quantum cannot be cloned,” *Nature (London)*, vol. 299, pp. 802–803, 1982.
- [3] P. Shor, “Algorithms for quantum computation: Discrete log and factoring,” *Proceedings of the 35th Annual Symposium on . . .*, Jan 1994.
- [4] L. Grover, “A fast quantum mechanical algorithm for database search,” *Proceedings of the twenty-eighth annual ACM . . .*, Jan 1996.
- [5] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, Jan 1978. RSA. BB88.
- [6] C. Bennett and G. Brassard, “Quantum cryptography: Public key distribution and coin tossing,” *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, vol. 175, 1984.
- [7] D. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.
- [8] R. Gallager, *Low-density parity-check codes*. PhD thesis, Massachusetts Institute of Technology, 1963.
- [9] G. Brassard and L. Salvail, “Secret key reconciliation by public discussion,” *Lecture Notes in Computer Science*, vol. 765, pp. 410–423, 1994.
- [10] S. Chung, G. F. Jr, and T. Richardson, “On the design of low-density parity-check codes within 0.0045 db of the shannon limit,” *IEEE Communications letters*, Jan 2001.
- [11] C. Elliott, A. Colvin, D. Pearson, and O. Pikalo, “Current status of the darpa quantum network,” *Arxiv preprint quant-ph*, Jan 2005.
- [12] J. Lodewyck, M. Bloch, R. García-Patrón, and S. Fossier, “Quantum key distribution over 25km with an all-fiber continuous-variable system,” *Phys. Rev. A*, Jan 2007.
- [13] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, 1948.

- [14] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [15] G. Vernam, “Cipher printing telegraph systems,” *Journal of the American Institute of Electrical Engineers*, 1926.
- [16] C. E. Shannon, “Communication theory of secrecy systems,” *Bell System Technical Journal*, vol. 28, pp. 656–715, 1949.
- [17] J. Carter and M. Wegman, “Universal classes of hash functions,” *Journal of computer and system sciences*, vol. 18, no. 2, pp. 143–154, 1979.
- [18] M. Wegman and J. Carter, “New hash functions and their use in authentication and set equality,” *Journal of computer and system sciences*, vol. 22, no. 3, pp. 265–279, 1981.
- [19] D. Bruß, D. DiVincenzo, A. Ekert, and C. Fuchs, “Optimal universal and state-dependent quantum cloning,” *Phys. Rev. A*, Jan 1998.
- [20] H. Bechmann-Pasquinucci and N. Gisin, “Incoherent and coherent eavesdropping in the six-state protocol of quantum . . .,” *Phys. Rev. A*, Jan 1999.
- [21] C. Bennett, “Quantum cryptography using any two nonorthogonal states,” *Physical Review Letters*, Jan 1992.
- [22] A. Ekert, “Quantum cryptography based on bell’s theorem,” *Physical Review Letters*, vol. 67, no. 6, pp. 661–663, 1991.
- [23] P. Shor and J. Preskill, “Simple proof of security of the bb84 quantum key distribution protocol,” *Physical Review Letters*, vol. 85, no. 2, pp. 441–444, 2000.
- [24] A. Einstein, B. Podolsky, and N. Rosen. . . , “Can quantum-mechanical description of physical reality be considered complete?,” *Physical review*, Jan 1935.
- [25] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dusek, N. Lutkenhaus, and M. Peev, “The security of practical quantum key distribution,” *arXiv*, vol. quant-ph, Jan 2008.
- [26] C. Bennett, G. Brassard, and J. Robert, “Privacy amplification by public discussion,” *SIAM Journal on Computing*, vol. 17, p. 210, 1988.
- [27] H. Lo and Y. Zhao, “Quantum cryptography,” *Arxiv preprint arXiv:0803.2507*, Jan 2008.

- [28] W. T. Buttler, S. K. Lamoreaux, J. R. Torgerson, G. H. Nickel, C. H. Donahue, and C. G. Peterson, "Fast, efficient error reconciliation for quantum cryptography," *Phys. Rev. A*, vol. 67, p. 052303, May 2003.
- [29] S. Liu, H. V. Tilborg, and M. V. Dijk, "A practical protocol for advantage distillation and information reconciliation," *Designs*, Jan 2003.
- [30] S. Rass and C. Kollmitzer, "Improved error correction in quantum key distribution protocols," *secoqc.net*, 2001.
- [31] T. Sugimoto and K. Yamazaki, "A study on secret key reconciliation protocol cascade", " ... AND COMPUTER SCIENCES E SERIES A, Jan 2000.
- [32] C. Bennett, G. Brassard, and C. Crepeau, "Generalized privacy amplification," *IEEE Transactions on Information Theory*, Jan 1995.
- [33] X. Ma, C. Fung, J. Boileau, and H. Chau, "Practical post-processing for quantum-key-distribution experiments," *Imprint*, Jan 2009.
- [34] R. Renner and R. König, "Universally composable privacy amplification against quantum adversaries," *Theory of Cryptography Conference (TCC)*, 2005.
- [35] "ID Quantique SA." <<http://www.idquantique.com/>>. [Accessed: 20.04.2010].
- [36] "MagiQ." <<http://www.magiqtech.com/>>. [Accessed: 20.04.2010].
- [37] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Reviews of Modern Physics*, Jan 2002.
- [38] W. Hwang, "Quantum key distribution with high loss: Toward global secure communication," *Physical Review Letters*, Jan 2003.
- [39] H. Lo, X. Ma, and K. Chen, "Decoy state quantum key distribution," *Physical Review Letters*, Jan 2005.
- [40] V. Makarov, A. Anisimov, and J. Skaar, "Effects of detector efficiency mismatch on security of quantum cryptosystems," *Phys. Rev. A*, Jan 2006.
- [41] B. Qi, C. Fung, H. Lo, and X. Ma, "Time-shift attack in practical quantum cryptosystems," *Arxiv preprint quant-ph*, Jan 2005.
- [42] L. Lydersen and J. Skaar, "Security of quantum key distribution with bit and basis dependent detector flaws," *arXiv*, vol. quant-ph, Jan 2008.

- [43] R. Renner, “Security of quantum key distribution,” *Arxiv preprint quant-ph*, Jan 2005.
- [44] H. Lo and H. Chau, “Unconditional security of quantum key distribution over arbitrarily long distances,” *Science*, Jan 1999.
- [45] H. Lo, “Method for decoupling error correction from privacy amplification,” *arXiv*, vol. quant-ph, Jan 2002.
- [46] D. Mayers, “Quantum key distribution and string oblivious transfer in noisy channels,” *Advances in Cryptology—CRYPTO’96*, Jan 1996.
- [47] M. Koashi and J. Preskill, “Secure quantum key distribution with an uncharacterized source,” *Physical Review Letters*, Jan 2003.
- [48] M. Koashi, “Simple security proof of quantum key distribution based on complementarity,” *New Journal of Physics*, Jan 2009.
- [49] H. Maassen and J. Uffink, “Generalized entropic uncertainty relations,” *Physical Review Letters*, Jan 1988.
- [50] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.
- [51] S. J. Johnson, “Introducing low-density parity-check codes,” *ACoRN Spring School Curriculum*, pp. 1–83, Apr 2008.
- [52] I. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *Journal of the Society for Industrial and Applied . . .*, Jan 1960.
- [53] C. Berrou, A. Glavieux, and P. Thitimajshima. . . , “Near shannon limit error correcting coding and decoding: Turbo-codes (1),” *IEEE International Conference on Communications*, Jan 1993.
- [54] Otmani, “On the minimum distance of generalized ldpc codes,” *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pp. 751 – 755, 2007.
- [55] Xiao-Yu, “On the computation of the minimum distance of low-density parity-check codes,” *Communications, 2004 IEEE International Conference on*, vol. 2, pp. 767 – 771 Vol.2, 2004.
- [56] F. Daneshgaran, M. Laddomada, and M. Mondin, “An algorithm for the computation of the minimum distance of ldpc codes,” *European Transactions on Telecommunications*, vol. 17, no. 1, pp. 57–62, 2006.

- [57] R. Tanner, “A recursive approach to low complexity codes,” *IEEE Transactions on Information Theory*, Jan 1981.
- [58] B. Bollobás, “Modern graph theory,” *books.google.com*, Jan 1998.
- [59] M. Luby, M. Mitzenmacher, and M. Shokrollahi, “Practical loss-resilient codes,” *Proceedings of the . . .*, Jan 1997.
- [60] M. Luby, M. Mitzenmacher, and M. Shokrollahi, “Analysis of low density codes and improved designs using irregular graphs,” *Proc. of ACM STOC*, Jan 1998.
- [61] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, “Improved low-density parity-check codes using irregular graphs,” *IEEE transactions on Information Theory*, vol. 47, no. 2, pp. 585–598, 2001.
- [62] D. MacKay, S. Wilson, and M. Davey, “Comparison of constructions of irregular gallager codes,” *IEEE Transactions on Communications*, Jan 1999.
- [63] T. Richardson, M. Shokrollahi, and R. Urbanke, “Design of capacity-approaching irregular low-density parity-checkcodes,” *IEEE Transactions on Information Theory*, Jan 2001.
- [64] X. Hu, E. Eleftheriou, and D. Arnold, “Progressive edge-growth tanner graphs,” *GLOBECOM-NEW YORK-*, vol. 2, pp. 995–1001, 2001.
- [65] X. Hu, E. Eleftheriou, and D. Arnold, “Regular and irregular progressive edge-growth tanner graphs,” *IEEE Transactions on Information . . .*, Jan 2005.
- [66] T. Richardson and R. Urbanke, “Efficient encoding of low-density parity-check codes,” *IEEE transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, 2001.
- [67] T. Richardson and R. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” *IEEE transactions on Information Theory*, Jan 2001.
- [68] M. Fossorier, M. Mihaljevic, and H. Imai, “. . . iterative decoding of low-density parity check codes based on belief propagation,” *IEEE Transactions on Communications*, Jan 1999.
- [69] L. Bazzi, T. Richardson, and R. Urbanke, “Exact thresholds and optimal codes for the binary-symmetric channel and gallager’s decoding algorithm a,” *IEEE transactions on Information Theory*, vol. 50, no. 9, pp. 2010–2021, 2004.

- [70] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE transactions on Information Theory*, Jan 1973.
- [71] A. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using ldpc codes," *IEEE Communications letters*, vol. 6, pp. 1–3, Jul 2002.
- [72] J. Chen, A. Dholakia, and E. Eleftheriou, "Reduced-complexity decoding of ldpc codes," *IEEE Transactions on Communications*, Jan 2005.
- [73] T. Zhang, Z. Wang, and K. Parhi, "On finite precision implementation of low density parity check codes decoder," *IEEE International Symposium on Circuits . . .*, Jan 2001.
- [74] D. Elkouss, A. Leverrier, R. Alléaume, and J. Boutros, "Efficient reconciliation protocol for discrete-variable quantum key distribution," *Arxiv preprint arXiv:0901.2140*, Jan 2009.
- [75] "LTHC : LDPC Optimization," February 2009. Information Processing Group, Ecole Polytechnique Federale de Lausanne <<http://ipgdemos.epfl.ch/ldpcopt/>>. [Accessed: 20.04.2010].
- [76] "LDPC Codes," January 2009. Signal Processing Microelectronics: University of Newcastle, Australia <<http://sigpromu.org/ldpc/>>. [Accessed: 20.04.2010].
- [77] J. L. Duligall, M. S. Godfrey, K. A. Harrison, W. J. Munro, and J. G. Rarity, "Low cost and compact quantum cryptography," *arXiv*, vol. quant-ph, Jan 2006.
- [78] R. Matsumoto, C. Schulte, G. Tack, and J. B. . . . , "Problems in application of ldpc codes to information reconciliation in quantum key . . . ," *Arxiv preprint arXiv:0908.2042*, Jan 2009.
- [79] V. Makarov, "Personal communication.," 22. april 2010.
- [80] I. Lucio-Martinez, P. Chan, X. Mo, S. Hosier, and W. Tittel, "Proof-of-concept of real-world quantum key distribution with quantum frames," *New Journal of Physics*, Jan 2009.
- [81] D. Pearson, "High-speed qkd reconciliation using forward error correction," *AIP Conference Proceedings*, vol. 734, no. 1, pp. 299–302, 2004.
- [82] A. Blad, O. Gustafsson, and L. Wanhammar, "A hybrid early decision-probability propagation decoding algorithm for low . . . ," *Signals*, Jan 2005.

- [83] N. Lütkenhaus, “Estimates for practical quantum cryptography,” *Phys. Rev. A*, Jan 1999.
- [84] D. Gottesman, N. Lütkenhaus, and J. Preskill, “Security of quantum key distribution with imperfect devices,” *arXiv*, vol. quant-ph, Dec 2002.
- [85] V. Scarani and R. Renner, “Quantum cryptography with finite resources: unconditional security bound for discrete- . . .,” *Physical Review Letters*, Jan 2008.



# Appendices

## A Codes

Table 3: Rate 0.50

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.397706000	$r_{10} =$	0.5241810
$b_3 =$	0.349409000	$r_{12} =$	0.4301330
$b_4 =$	0.025287000	$r_{13} =$	0.0250194
$b_5 =$	0.002567270	$r_{14} =$	0.0206666
$b_6 =$	0.043127500		
$b_8 =$	0.037147500		
$b_9 =$	0.034218000		
$b_{10} =$	0.007106810		
$b_{11} =$	0.000811348		
$b_{14} =$	0.024236300		
$b_{15} =$	0.048168400		
$b_{17} =$	0.004799440		
$b_{47} =$	0.001030330		
$b_{49} =$	0.000730806		
$b_{55} =$	0.000210349		
$b_{56} =$	0.000097394		
$b_{57} =$	0.010803700		
$b_{58} =$	0.005925190		
$b_{59} =$	0.004756550		
$b_{66} =$	0.001861430		
$p_{SL} = 0.11003$	$p_{DEL} = 0.1071$ [74]	$p_{OL} = 0.096$	$\kappa(p_{OL}) = 1.0960$

Table 4: Rate 0.55

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.4067190	$r_{10} =$	0.295769
$b_3 =$	0.3372460	$r_{11} =$	0.704231
$b_6 =$	0.1453320		
$b_{15} =$	0.0123558		
$b_{16} =$	0.0079362		
$b_{18} =$	0.0627909		
$b_{19} =$	0.0147485		
$b_{31} =$	0.0128713		
$p_{SL} = 0.09410$	$p_{DEL} = 0.0904$ [74]	$p_{OL} = 0.083$	$\kappa(p_{OL}) = 1.0905$

Table 5: Rate 0.60

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.45980692549363	$r_8 =$	1
$b_3 =$	0.34829693138697		
$b_6 =$	0.10777718984060		
$b_7 =$	0.08411895327880		
$p_{SL} = 0.07938$	$p_{DEL} = 0.0740$ [75]	$p_{OL} = 0.063$	$\kappa(p_{OL}) = 1.1658$

Table 6: Rate 0.63

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.41719537981987	$r_{10} =$	0.20510304219823
$b_3 =$	0.34050103405756	$r_{11} =$	0.79489695780177
$b_6 =$	0.12600902776679		
$b_7 =$	0.03513697839130		
$b_{14} =$	0.08115757996448		
$p_{SL} = 0.07110$	$p_{DEL} = 0.0710$ [75]	$p_{OL} = 0.061$	$\kappa(p_{OL}) = 1.1165$

Table 7: Rate 0.65

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.3863726069793800	$r_{14} =$	1
$b_3 =$	0.3252066829641300		
$b_6 =$	0.0265284543109450		
$b_7 =$	0.1389836924391200		
$b_8 =$	0.0494315391272960		
$b_{22} =$	0.0667710080462730		
$b_{23} =$	0.0066216726649948		
$b_{34} =$	0.0000843434678622		
$p_{SL} = 0.06579$	$p_{DEL} = 0.0630$ [75]	$p_{OL} = 0.058$	$\kappa(p_{OL}) = 1.0956$

Table 8: Rate 0.68

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.381074366111340	$r_{15} =$	0.51612903225806
$b_3 =$	0.348601162110540	$r_{16} =$	0.48387096774194
$b_6 =$	0.042424754987559		
$b_7 =$	0.140111586299560		
$b_{14} =$	0.049870123655895		
$b_{32} =$	0.037918006835106		
$p_{SL} = 0.05814$	$p_{DEL} = 0.0580$ [75]	$p_{OL} = 0.051$	$\kappa(p_{OL}) = 1.1011$

Table 9: Rate 0.70

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.3693962887197200	$r_{17} =$	0.3121387283237
$b_3 =$	0.3550992114952100	$r_{18} =$	0.6878612716763
$b_7 =$	0.1911809886586900		
$b_{12} =$	0.0059314386693899		
$b_{18} =$	0.0504218275269760		
$b_{32} =$	0.0070440808902666		
$b_{42} =$	0.0108114745096090		
$b_{50} =$	0.0101146895301310		
$p_{SL} = 0.05324$	$p_{DEL} = 0.0530$ [75]	$p_{OL} = 0.047$	$\kappa(p_{OL}) = 1.0968$

Table 10: Rate 0.72

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.3640227133735600	$r_{17} =$	0.51428571428571
$b_3 =$	0.3671055632144600	$r_{18} =$	0.48571428571429
$b_7 =$	0.1371867013486300		
$b_8 =$	0.0682772087712480		
$b_{24} =$	0.0515030947547050		
$b_{26} =$	0.0046458171742165		
$b_{28} =$	0.0072589013631725		
$p_{SL} = 0.0485$	$p_{DEL} = 0.0480$ [75]	$p_{OL} = 0.0420$	$\kappa(p_{OL}) = 1.1138$

Table 11: Rate 0.75

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.3354502837337800	$r_{24} =$	1
$b_3 =$	0.2958553682783500		
$b_6 =$	0.0721405459627580		
$b_7 =$	0.2112209775643500		
$b_9 =$	0.0014214036316315		
$b_{30} =$	0.0839114208291380		
$p_{SL} = 0.0417$	$p_{DEL} = 0.0410$ [75]	$p_{OL} = 0.037$	$\kappa(p_{OL}) = 1.0947$

Table 12: Rate 0.80

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.28541565263977	$r_{35} =$	1
$b_3 =$	0.46250126583224		
$b_{20} =$	0.25208308152800		
$p_{SL} = 0.0311$	$p_{DEL} = 0.0270$ [75]	$p_{OL} = 0.024$	$\kappa(p_{OL}) = 1.1858$

Table 13: Rate 0.85

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.2710259552986300	$r_{41} =$	1
$b_3 =$	0.3521523317173300		
$b_7 =$	0.2609445315165000		
$b_{16} =$	0.0508864229335420		
$b_{17} =$	0.0045071689515054		
$b_{23} =$	0.0199496014006440		
$b_{28} =$	0.0360195339293610		
$b_{48} =$	0.0045144542524856		
$p_{SL} = 0.0215$	$p_{DEL} = 0.0200$ [75]	$p_{OL} = 0.018$	$\kappa(p_{OL}) = 1.1533$

Table 14: Rate 0.90

Column Distribution, $b$		Row Distribution, $r$	
$b_2 =$	0.1925270000000000	$r_{50} =$	0.951178
$b_3 =$	0.4690030000000000	$r_{51} =$	0.048822
$b_5 =$	0.0894705000000000		
$b_9 =$	0.1091710000000000		
$b_{12} =$	0.1278300000000000		
$b_{21} =$	0.0119974000000000		
$p_{SL} = 0.0130$	$p_{DEL} = 0.0109$ [74]	$p_{OL} = 0.009$	$\kappa(p_{OL}) = 1.3500$

## B Source Code

### List of Listings

1	Check Node Object . . . . .	77
2	Horizontal Update . . . . .	78
3	Parity Check . . . . .	79

Listing 1: Check Node Object

```
/*
A class that simulates a check node from a Tanner-graph [Tanner 1981]. Contains private
information on which variable nodes it is connected to and which edge number in the
variable node each edge corresponds to. In that way the horizontal step in an LLR-based
BP-algorithm [Chen 2005] can be executed within each node, performed by the function
computeHorizontalBCJR() or computeHorizontalMIN() and the resulting extrinsic messages
can be directly forwarded to the involved variable nodes.
*/

class CheckNode
{
private:
// Contains the updated number of edges during construction, and finalized number of edges
during decoding
int nEdges_;
// Used in Construction only. Contains the maximum number of edges during construction.
int nEdgeMax_;

// Edges. Edge #i is fully described by the contents of
// the i'th entry in these three vectors

// nvEdgeCoords_[i] contains the bit-number to which edge #i is connected.
vector<int> nvEdgeCoords_;
// nvEdgeCoordsLocation_[i] contains the edge number of edge #i in the variable node it is
connected to.
vector<int> nvEdgeCoordLocation_;
// Used in decoding only. dvMessageIn_[i] Contains the incoming messages along edge #i.
vector<double> dvMessageIn_;

// Used in decoding only. Contains the 'true' parity of this check node, as received from
Alice.
int nTrueParity_;
// Used in construction only. Denotes which tier in a tree expansion this node lies
int nDepth_;

public:

//// CONSTRUCTOR ////

// Simply initiate all variables to 0 when an instance is created.
CheckNode(){
nEdges_ = 0;
nEdgeMax_ = 0;
nTrueParity_ = 0;
nDepth_ = 0;
}

//// GETTERS ////

// Returns the current number of edges of this node. Used in both construction and decoding.
int getEdges(){ return nEdges_; }

// Returns the coordinate of the variable node to which edge #nEdgeNo is connected to
int getCoord(int nEdgeNo){ return nvEdgeCoords_[nEdgeNo]; }

// Returns the depth of this node in a tree expansion.
int getDepth(){ return nDepth_; }

// Tests if the number of current edges equals the maximum amount. Used in construction only.
bool isMaxed(){
if(nEdges_>=nEdgeMax_){ return true; }
else{ return false; }
}
}
```

```

//// SETTERS ////

// Sets the maximum number of edges this node can have. Used once in construction.
void setMax(int nMax){ nEdgeMax_ = nMax; }

// Used in the initialization of the decoding algorithm only.
// Sets the internal 'true' parity equal to that observed by Alice.
void setParity(int nTrueParity){ nTrueParity_ = nTrueParity; }

// Used in decoding only, by the variable nodes this node is connected to.
// Enabling them to directly set the incoming messages to this node
void pushMessage(int nEdgeNo, double dMessage){
    if(nEdgeNo<nEdges_){ dvMessageIn_[nEdgeNo] = dMessage; }
}

// Used in construction only. When expanding a tree this functions sets the depth
// when this node is encountered.
void setDepth(int nDepth){ nDepth_ = nDepth; }

//// FUNCTIONS ////

// Used in construction (and retrieval from file). Adds an edge from this node to variable
// node
// #nVariableNo. nEdgeLocation is the number of this edge in the variable node.
void addEdge(int nVariableNo, int nEdgeLocation);

// Used by Alice to calculate the syndrome which she send to Bob, prior to decoding.
int produceSyndrome(vector< VariableNode > &variables);

// Used in decoding to check if the hard decisions made in the variable nodes
// satisfy the true parity of this node.
bool parityCheck(vector< VariableNode > &variables);

// Used in decoding only. The main horizontal step where the incoming messages
// are evaluated and the new outgoing messages are produced.
void computeHorizontal(vector< VariableNode > &variables);

// Used in decoding only. An approximation to the numerically accurate
// computeHorizontalBCJR() operation. Can increase computation speed
// at the cost of decoding probability, with suitably chosen normalization factor dAlpha
// [Chen 2005].
void computeHorizontalMIN(vector< VariableNode > &variables, double dAlpha);
};

```

## Listing 2: Horizontal Update

```

void CheckNode::computeHorizontal(vector< VariableNode > &variables){
    int nEdgeCounter, nTempSign;
    double dTempTanh;

    vector<double> dvTanhMessage(nEdges_);
    for(nEdgeCounter=0;nEdgeCounter<nEdges_ ;nEdgeCounter++){
        dvTanhMessage[nEdgeCounter] = (double) tanh(dvMessageIn_[nEdgeCounter] / 2.0);
    }

    // FORWARD-BACKWARD-PASS //

    vector<double> dvForwardPass(nEdges_+1);
    vector<double> dvBackwardPass(nEdges_+1);

    dvForwardPass[0] = 1;
    dvBackwardPass[nEdges_] = 1;

    // Forward Pass //
    // Loops through all the edges and calculates for each edge the product of the incoming
    // messages from
    // all the preceding edges. dvForwardPass[nEdges-1] is omitted since it would contain the
    // product of all
    // the messages and is thus useless in the ensuing combination step.
    for(nEdgeCounter=0;nEdgeCounter<nEdges_-1;nEdgeCounter++){
        dvForwardPass[nEdgeCounter+1] = dvForwardPass[nEdgeCounter] * dvTanhMessage[nEdgeCounter];
    }

    // Backward Pass //
    // Loops backward through all the edges and calculates for each edge the product of the
    // messages
    // from all the 'higher' edges. dvBackwardPass[0] is omitted since it would contain the
    // product of all

```

```

// the messages and is thus useless in the ensuing combination step.
for (nEdgeCounter=nEdges_-1;nEdgeCounter>0;nEdgeCounter--){
    dvBackwardPass[nEdgeCounter] = dvBackwardPass[nEdgeCounter+1] * dvTanhMessage[nEdgeCounter];
}

// Combination //
for (nEdgeCounter=0;nEdgeCounter<nEdges_;nEdgeCounter++){ // Looping through all the edges.

    // By multiplying the forward and backward pass together in this way, the message (dTempTan)
    // now
    // consists of the the product of the tanh() of all the incoming messages except the one
    // along nEdgeCounter.
    dTempTanh = dvForwardPass[nEdgeCounter] * dvBackwardPass[nEdgeCounter+1];

    // The outgoing message from this node along edge number #nEdgeCounter is defined as the
    // logarithm
    // of the probability that this node has the correct parity (traditionally 0), assuming that
    // all the
    // incoming messages along the other edges are correct, divided by the analogue probability
    // of the incorrect parit (traditionally 1). However in our implementation, the correct
    // parity might
    // be 1 if that is what Alice observed. In these cases the probability of a correct and
    // incorrect
    // parity are interchanged, resulting in a sign-flip on the LLR-message.
    nTempSign = (nTrueParity_ == 1) ? -sign(dTempTanh) : sign(dTempTanh);

    dTempTanh = abs(dTempTanh);

    // Making sure that the messages don't escalate infinitely.
    if (dTempTanh > D_MAX_TANH){
        variables[nvEdgeCoords_[nEdgeCounter]].pushMessage(nvEdgeCoordLocation_[nEdgeCounter] ,
            nTempSign * D_MAX_ARCTANH);
    }
    else{
        variables[nvEdgeCoords_[nEdgeCounter]].pushMessage(nvEdgeCoordLocation_[nEdgeCounter] ,
            2 * nTempSign * atanh(dTempTanh));
    }

    // Exporting the message along edge number #nEdgeCounter to variable number
    // #nvEdgeCoords_[nEdgeCounter]. The final message is the product
    // 2*(correct sign)*arctanh(product of tanh(incoming messages)) as described by Chen.
}
}

```

### Listing 3: Parity Check

```

// Used by Bob in decoding.
// After a vertical step all the variable nodes have updated their hard decisions on bit
// values.
// This function XOR's together the hard decisions and evaluates if the resulting parity equals
// the true parity sent by Alice, and returns the corresponding boolean.

bool CheckNode::parityCheck(vector< VariableNode > &variables){

    int nParity = 0;

    // Loops through all the edges of this node.
    for (int nEdgeCounter=0;nEdgeCounter < nEdges_;nEdgeCounter++){
        // XOR's the hard decisions together.
        nParity ^= variables[nvEdgeCoords_[nEdgeCounter]].getHardDecision();
    }

    // Evaluates the current parity of this node to the true one from Alice.
    if (nParity == nTrueParity_){ return true; }
    else{ return false; }
}

```