Implementation of a Quantum Key Distribution Protocol from an Entangled Photon Detection Setup on Embedded Devices

Lukas Plazovnik

BACHELORARBEIT

eingereicht am
Fachhochschul-Bachelorstudiengang

Hardware-Software-Design

in Hagenberg

im August 2015

Diese Arbeit entstand im Rahmen des Gegenstandes

Seminar zur Bachelorarbeit

im

Sommersemester 2015

Betreuer:

FH-Prof. DI Michael Bogner
Dr. Rupert Ursin

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, August 7, 2015

Lukas Plazovnik

Contents

Declaration iii

Kurzfassung vi

Abstract vii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Objective . . . 2
- 1.3 Structure . . . 2

2 Quantum Mechanics Introduction 3
- 2.1 The Photon: Particle or Wave? . . . 3
- 2.2 Polarization . . . 6
- 2.3 Uncertainty Principle . . . 7
- 2.4 No-cloning Theorem . . . 9
- 2.5 Entanglement . . . 10
- 2.6 Bell Measurement of Polarization . . . 10

3 Quantum Key Distribution Protocol 12
- 3.1 Classical Encryption . . . 12
- 3.2 Quantum Protocol . . . 13
- 3.3 BB84 Protocol . . . 15
- 3.4 Entanglement Protocol . . . 17
- 3.5 Summary . . . 18

4 Synchronization 19
- 4.1 Clock Accuracy and Stability . . . 19
- 4.2 Coincidences . . . 21
- 4.3 Hardware Synchronization . . . 22
- 4.4 Software Synchronization . . . 23
- 4.5 Summary . . . 24

5 Quantum Key Encryption 25

5.1 Motivation . . . 25
5.2 Overview . . . 26
5.3 Scenarios . . . 27
5.4 The Source . . . 28
5.5 Bell Measurement . . . 29
5.6 The System-on-Chip . . . 29

6 Hardware **32**
6.1 Zedboard . . . 32
6.2 Time-Tagging Module . . . 34
6.3 GPS Module . . . 34

7 Software **36**
7.1 Operating System . . . 36
7.2 Austrian Institute of Technology Quantum Key Distribution . 37

8 Verification **41**
8.1 Motivation . . . 41
8.2 Commissioning . . . 41
8.2.1 Changes . . . 41
8.2.2 Source . . . 41
8.2.3 Bell Measurement . . . 42
8.2.4 System-on-Chip . . . 42
8.3 Linux Distribution . . . 42
8.4 Time-Tagging Module Testing . . . 43
8.5 QKD Module Testing . . . 44
8.6 Summary . . . 45

9 Conclusion and Outlook **46**

References **48**
Literature . . . 48
Online sources . . . 50

Kurzfassung

Informationssicherheit ist eines der heikelsten Themen unserer Zeit. Es wird immer einfacher, große Datenmenge zu speichern und auszuwerten. Für sensible Anwendungen wie Banktransaktionen, Geschäftsmeetings oder auch militärische Aktionen, ist es wichtig, dass diese nicht von Dritte abgehört werden können. Die aktuelle Situation ist allerdings, dass unsere gängigsten Verschlüsselungssysteme unsicher, und lediglich durch fehlende Algorithmen für die Zerlegung von Primfaktoren nicht in relevanter Zeit entschlüsselbar sind. Sollte ein solcher Algorithmus gefunden werden sind große Teile unseres aktuellem Sicherheits Netzwerkes öffentlich zugänglich.

In dieser Arbeit wird gezeigt, welche Möglichkeit es jenseits klassischer Verschlüsselungsysteme noch gibt. Der Quanten-Schlüsselaustausch ist eine Option um aus quantenmechanischen Effekten eine Verschlüsselung zu erzeugen um auch zukünftig eine sichere Kommunikation zu gewährleisten. Die Quantenmechanik liefert hier eine Möglichkeit jeden Abhörversuch eines Dritten zu erkennen und entsprechend darauf zu reagieren. Der Quanten-Schlüsselaustausch ist theoretisch sicher, allerdings werden in realer Implementierung aufgrund von Informationsverlust und Messfehlern zusätzliche Nacharbeiten erledigt, für die Software benötigt wird.

Aktuelle Implementierungen dieses Verfahrens sind aufwendig und sehr unausgereift. So wird der gesamte Prozess in viele kleinen Abschnitten aufgeteilt, für die auch verschiedene Hardware und Know-how benötigt wird. Um diese Technologie zu fördern und zu vereinfachen, wurde ein System entwickelt, welches analoge Signale messen und verarbeiten kann. Dazu wird eine Software des Austrian Institute of Technology auf ein Eingebettetes System portiert. Da ein Quanten-Schlüsselaustausch dann nur mehr auf einer Plattform geschieht, fallen Anschaffungskosten und Einarbeitungszeit weg.

Abstract

Security of information is one of the trickiest topics of our time. Modern information systems are capable of analyzing a huge amount of data very quickly. To prevent a third party from stealing sensitive information, it is necessary to encrypt communication. The problem is that none of our common encryption systems is totally secure, or at least it will not be in the future. Only the absence of a fast algorithm for prime factorization protects us from being intercepted at transferring data on bank transactions, business meetings or military operations.

This thesis presents an alternative solution beside classical cryptography. Quantum key distribution (QKD) is a method to create a key for encryption out of quantum mechanical properties. The created key ensures that communication will be still secure in the future. Quantum mechanical properties deliver a method to detect any interception during the communication and react accordingly. In theory, QKD is completely secure although it needs additional processes which have to be handled due to practical and measuring errors.

Current implementations of quantum key distribution are complex and hard to handle. The process has to be divided into several single steps and require certain hardware and know how. To boost this technology and make it more easily available, a module was developed which is capable of measuring the raw key and of post processing the final key for encryption. Additionally, single process layers were structured in such a way, that future modifying is possible and single parts of the module can be easily replaced. For this purpose the quantum key distribution software of the Austrian Institute of Technology is ported to a System-on-Chip device. With this device, quantum key distribution can run on one single device and therefore easier to use.

Chapter 1

Introduction

1.1 Motivation

There are numerous possibilities to collect and analyze data and the various ways to do so are still growing. With the introduction of big data and the internet of things more and more common applications will have a live connection to the internet. The information collected can be used in various ways to analyze behaviour of private parties, companies and societies, to foresee behaviours and habits. Although there are quite a lot positive examples, the collection of information can also be exploited and privacy can be violated. Today, for big telecommunication companies in order to foresee behaviour and habits. Although there are quite a lot of positive examples, the collection of information can also be exploited and privacy can be violated. To gain privacy in a communication between two partners, one has to encrypt messages to make sure no third person is listening to the conversation. Currently, the most common encryption is the RSA method. Although this method might be still secure, it probably will not persevere the next 50 years. Big IT-companies investing billions of dollars to build a quantum computer. Such an quantum computer capable of running Shor's Algorithm, would break RSA encryption in a very short time [17]. Considering the fast development of telecommunication, a new way of securing communication has been made, to guarantee privacy in the future. One way to achieve that, is to use a quantum key distribution system. This quantum key can be used for encrypting and decrypting a message and is totally secure in theory.

Charles Bennett and Gilles Brassard developed a quantum encryption protocol in 1984 for this issue [1]. A quantum encryption protocol manages to generate a key out of the law of quantum mechanics. In classical physics, in which current computers work, everything is deterministic. Encryption key generating processes are comprehensible and therefore can be recreated if enough information is collected. Recent studies have shown, that even from sound of CPUs, a key can be reconstructed [7]. A quantum key distribution

system provides a physically save method of generating keys which can not be reconstructed, even when all possible informations have been gathered. A Quantum Key Distribution protocol does not only present a solution to the encryption, but also the certainty that no third party has stolen the key.

1.2 Objective

Current implementations of Quantum Key Distribution systems are complex and hard to use. Many steps have to be prepared to start one single key generating process. If one want to change the location or the setup of the system. The whole process of calibrating an setting up has to be redone. The goal of this thesis is, to create a new abstraction layer for the distribution of a quantum key to promote and simplify future development in this area. Already know and proved processes shall be automated and simplified.

A quantum key distribution, theoretically secure, has insecurity in practical applications. By distributing a key to a specific location, information may get lost or wrong information can be detected. To compensate measurement and delivering errors, known algorithm and software can be used for compensation to regain security. The module developed in this thesis may detect this flaws and reconstruct the faulty key, although some information is lost. This module is capable to measure and create a raw key and post process all necessary steps to gain a usable key for secure communication. Also, single layers can be easily adjusted to speed up further research and testing of new hardware. Physicists do not have to post process their raw data manually, but can outsource their needed tasks to software, developed by computer scientist.

1.3 Structure

This thesis is structured as follows: In chapter 2, an introduction to important quantum mechanical effects will be made to understand the principle of quantum key distribution to gain a secure key and therefore secure communication. Chapter 3 and 4 give a further overview of the overall system for quantum key distribution. It explains the overall setup and commonly known protocols for QKD. Chapter 5 gives 2 different examples of how an implementation can be made. Chapter 6 and 7 gives the practical implementation of the system and serves as manual for this application in hardware and software. The chapter 8 and 9 presents results of the test applications and give an outlook.

Chapter 2

Quantum Mechanics Introduction

Quantum mechanics describes the physics and behavior of matter on an atomic scale. Quanta do not behave like waves, they do not behave like particles, they do not behave like clouds, or billiard balls, or weights on springs, or like anything that you have ever seen [6].
A photon, a quantum of light, is a packet which science can modify and observe. Thus it is no big surprise that the first quantum mechanics applications use photons. In this thesis a quantum key encryption (QKD) system will be introduced. A QKD system is a method for creating a key to en-/decrypt messages so no one else except the sender and the receiver can read it. Properties of light packages can be measured and out of these results a key for encrypting a message can be generated.

In classical physics observers are capable of reconstructing the key generating process if all relevant data is known. This is not possible in the principles of quantum mechanics. Quantum mechanics may have unaccustomed behaviours, but theses effect are proven and have been used for decades.

To understand how a quantum key distribution (QKD) system works, basic quantum mechanics effects have to be understood and thus will be explained in this chapter.

2.1 The Photon: Particle or Wave?

To foresee the behavior of light was almost impossible for a very long time. In the beginning of the 20th century, scientists discovered that the light behaves like particles and waves. These characteristics, however, were actually mutually exclusive until Einstein, Podolsky and Rosen published their paper "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?" in 1935 [4]. This section describes the behaviour of light, which is a important effect in quantum key distribution. To understand how the

light behaves, the Double-slit experiment will be examined.
The Double-slit experiment is one of the most famous experiments when it comes to describing the wave-particle duality in quantum mechanics.

First, light has to be considered a wave as seen in lakes or oceans. If a kid throws a stone into a lake bordered by a dam with two gaps, circular waves are produced as the stone hits the water surface.

As the stone hits the water surface. The waves extend circularly and only the part of the wave which hits the gaps gets through as shown in Figure 2.1a

Everything else will be reflected and will not be needed anymore for the wave consideration. The part of the wave which is split up by the two openings as shown in Figure 2.1b turned now to two new individual waves, interfering with each other, adding up, or subtracting themselves as shown in Figure 2.1c.

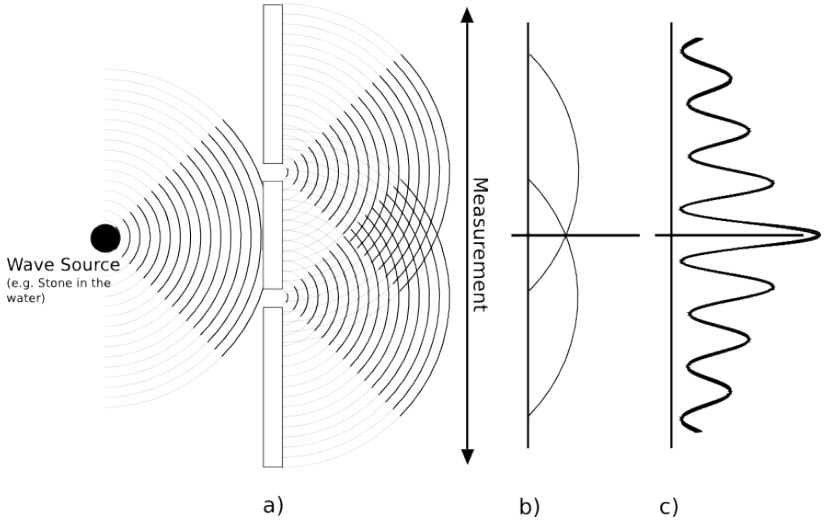

Figure 2.1: The light behaves like a wave. A big wave is approaching the slit and emits two new, individual waves. The two new waves, are either adding up or subtracting themselves. It is not possible to determine the location of the wave [6].

Depending on the circumstances, however, light does not behave like waves, but like particles. As a particle source, the kid appears again. It now continuously throws stones at a wall with two windows. The kid is also restricted by throwing only one stone at a time since it is using the other hand to pick them up. It throws the stones at a random direction. Sometimes it hits the wall and sometimes it gets a stone through the window. Every stone that gets through the windows stacks up and never subtracts as shown in Figure 2.2.

Until now, only theoretical assumptions have been made, but what does

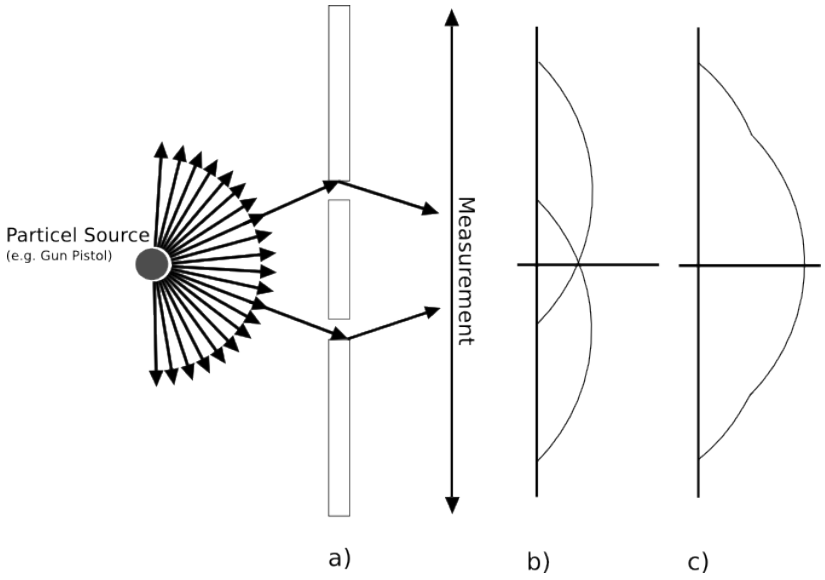

Figure 2.2: The light behaves like a particle. A single particle is emitted by the particle source and goes either through the top or the bottom slit. There is nothing else, the particle can interfere with. Hence, particles are stacking up and the location can be determined [6].

reality say about the behaviour of light? Is a photon a wave or a particle? As unlikely as the answer is, it behaves as a particle and as a wave.

If the Double-slit experiment is carried out under real conditions, the pattern which can be observed is the wave pattern. The wave interferes with itself but this assumption does not work well if the setup is changed a little bit. For scientists it is possible to provide a source which is capable of shooting a single photon at a time. By measuring the location of the photon, which could go either through the left slit or the right slit, the pattern will now show a classical particle pattern as shown in Figure 2.3. Again, the behaviour of a photon is changed just by observing it.

A photon, like any other quantum, behaves in a way, people are not used to. In daily life, objects have a specific position and a defined direction. This is not the case at nano scales. Photons behave as waves and as particles, and this has to be considered if one wants to understand the behaviour of the light. Furthermore, the wave-particle duality is necessary to understand, why QKD is secure.

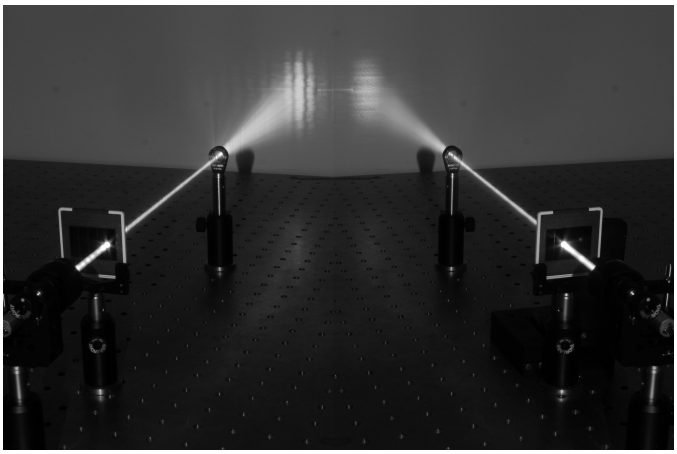

Figure 2.3: Left: wave pattern. Right: particle pattern [28]

2.2 Polarization

Polarization is a property of light which can hardly [18] be recognized by the human eye, although instruments exist that help us to distinguish it. In this section, an explanation is given for the polarization of light. Polarization is a very important effect in this thesis, since this property can be entangled and measured. Therefore, it is possible to obtain a quantum key.

Polarization can be seen as a kind of orientation of the electric field. Waves are mainly distinguished by their wavelength. A wavelength of 450nm can be recognized by the human eyes as the colour blue. Although our eyes do not differentiate polarizations, it can be measured and manipulated. To have an idea of polarization it will be explained in an example with a garden hose as shown in Figure 2.4: Moving the garden hose up and down produces a water wave with vertical (|V>) polarization, moving it left and right generates horizontal (|H>) polarization. As this can be done in any direction, the number of bases which can be measured is infinite. Although, basic quantum mechanics experiments are mostly fine with polarization in horizontal (|H>), vertical (|V>), plus (|+>) and minus (|->) bases as shown in Figure 2.5.

In nature, every possible polarization can be found, but in order to use it in our experiments we have to define bases in which polarization has to be measured. To achieve that, polarization filter, capable of blocking all waves which are not in a defined base can be used as shown in Figure 2.6.

Due to the Heisenberg uncertainty principle, it will be described in the next chapter, actually measuring the current state is almost impossible. Still, it is possible to filter polarization and do a Bell measurement.

2.3 Uncertainty Principle

Every object in the universe behaves like particles and waves. Due to this particle-wave duality it is not possible to simultaneously determine location and impulse of an object. Although this can be ignored in daily use, this can not be done with nano-scale objects. For example, an electron spinning around an atom: Again, neutrons behave like particles and waves, so both

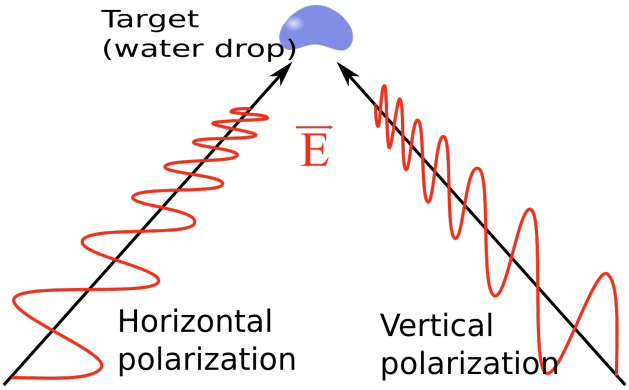

Figure 2.4: Model of polarization [27]

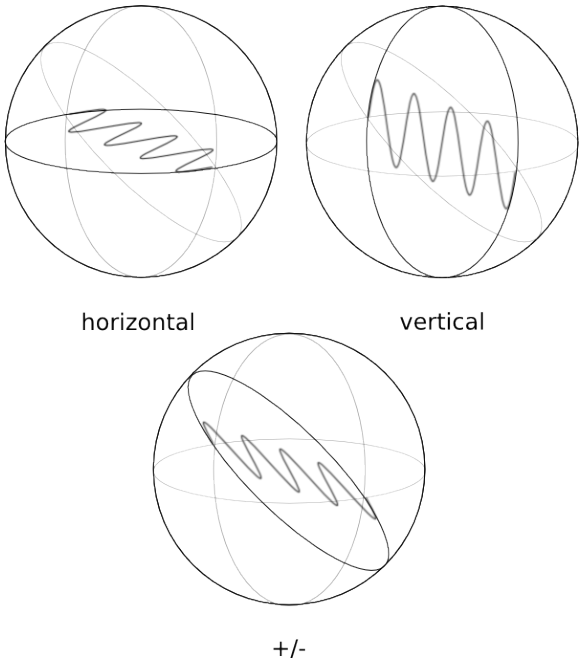

Figure 2.5: Common polarization bases used by physicists

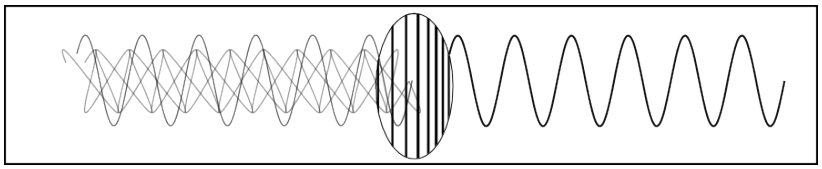

Figure 2.6: Model of polarization filtering

models will be presented and then merged. Particles can be pinned down to a specific location. The wave's exact location, however can not be specified but the wavelength, equivalent to momentum, can as Figure 2.7 explains.

Yet, we can define a wavelength within waves, which is equivalent to the momentum. This property is missing within a particle.

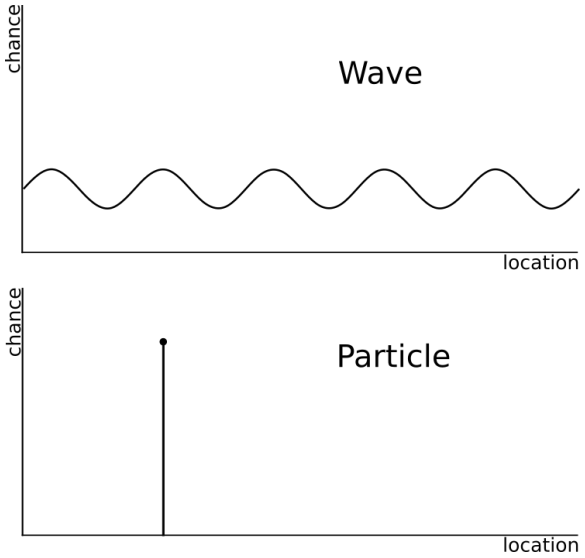

Figure 2.7: Probability of finding an object; wave: location can not be determined; particle: location is defined

To get more exact information on the location, a given wave can be interfered with additional, different waves. As a result, they will either add up or subtract. The result has higher peaks and more roots as shown in Figure 2.8b.

Higher peak means that the probability of finding the electron at this location rises. As the percentage of finding the electron at a specific location rises, information on the original wavelength gets lost. Hence, the wavelength is a sum of all wavelengths which were added to the original wavelength. Thus it can not be distinguished any more. Again, the more precise the location is known, the less accurate the information about the momentum

is. The less precise we know the location, the more precise we can measure the momentum. Figure 2.8c shows that.

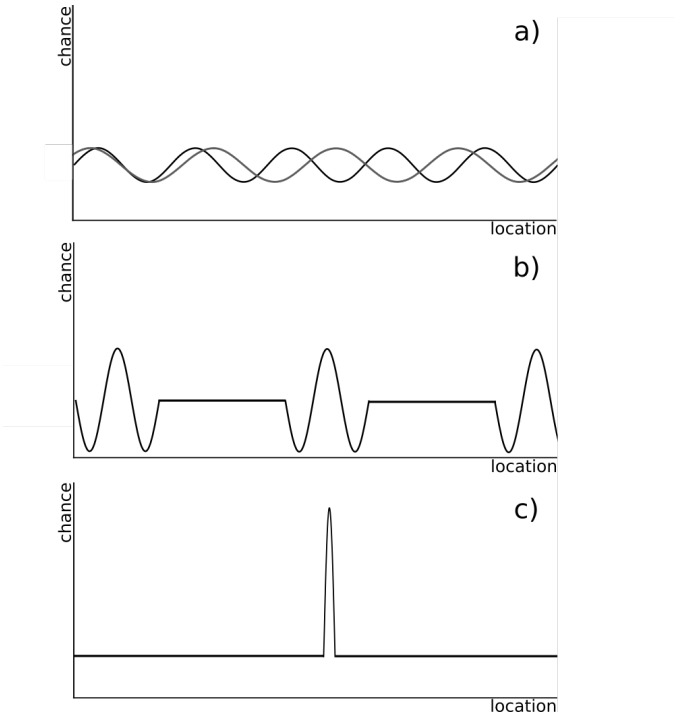

Figure 2.8: Since it is not possible to determine the location of the object in a wave, additional waves can be added to the original wave. By doing that, the possible number of locations decrease and eventually reaches one

This uncertainty principle is one of the reasons why quantum key distribution is more secure than traditional key distribution. Because of this effect, quantum bits can not be cloned.

2.4 No-cloning Theorem

In classical science, one of the most basic things is to copy an object or data. Especially in information science a file can be read and exact copies can be made. In 1982, Wootters and Zurek published a paper in which they explained that an exact copy of a quantum state can not be cloned; not because of missing instruments, but because of the law of physics: "If photons could be cloned, a plausible argument could be made for faster-than-light communication" [22].

2.5 Entanglement

Entanglement describes the non-classical correlation between dual or multi particles, with which they cannot be interpreted independently from each other [4].

2.6 Bell Measurement of Polarization

To measure a complete state of quantum is impossible due to the uncertainty principle, but it is possible to measure specific properties of a quantum at a time by doing a bell measurement. In the following section, a bell measurement for polarization will be explained. As with most of the principles shown, this can also be done for other properties of light.

To measure the event of a photon arriving, Single-Photon Avalanche Diodes (SPADs) are used for recognition. A SPAD can recognize a quantum event (an arriving photon) by a LVDS signal triggered by the photon. However, SPADs can not differ between |H> and |V> polarized photons. To differ between the 4 polarizations (|H>, |V> |+>, |->) 4 different paths are needed. Each path represents one polarization state. An arriving photon goes through a half-wave plate which is capable of rotating the polarization plane (|H>|V> or |+>|->) .

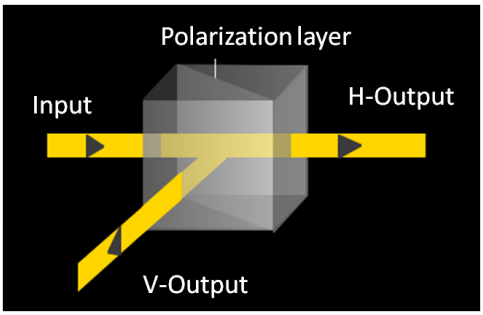

Figure 2.9: Polarized beam splitter [30]

After this is done, the photon paths are randomly splitted half-half by a beam splitter. The half wave plate decides the plane the operator wants to measure. The beam splitter then randomizes the measurement base. This is also a reason why quantum key distribution is secure. After that, a polarized beam splitter as shown in Figure 2.9, is used to route the path of photons in terms of polarization. In contrast to a standard beam splitter, which randomly routes the path, a polarized beam splitter dependently routes photons with polarization of |H> and |V>. Since it is hard to measure in |+> and |-> bases, the half-wave plate rotates |+> and |-> polarization by 45 degrees, into |H>/|V> plane. This is sketched in Figure 2.10.

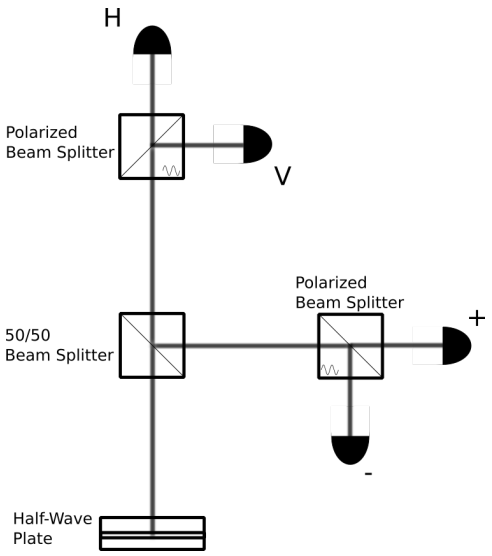

Figure 2.10: Bell measurement: half-wave plate chooses the planes, beam splitter randomly routes the photon, polarized beam splitter routes the final path to the detector.

As stated before, a bell measurement can be done in various way. One important fact is that the photon is destroyed by the measurement process. Thus, no second property can be measured. This bell measurement is a critical requirement for quantum key distribution and it will be explained in the next chapter.

Chapter 3

Quantum Key Distribution Protocol

3.1 Classical Encryption

Encryption has a long history since it has been traced back to the ancient Egypt and multiple other advanced cultures. Since then, every culture has a sort of description for hiding sensitive information. Julius Caesar, for example, encrypted his messages: "If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others" [14].
In the first years of telecommunication, encryption was mainly used for military, diplomatic and business purposes. But time has changed and almost every site on the web collects some sort of data about the person who visits its content. Today, encryption is closely connected to privacy, which, in the century of Big Data, is a very delicate topic. Encryption is the art of creating a message only readable by authenticated receivers. Hence, even if eavesdroppers managed to get hands on the message, they would not be able to read it [2].

There are practically two possibilities to encrypt a message: Asymmetric and symmetric. In an asymmetric encryption, the originator (Alice) uses a different key for encryption, than the key the recipient (Bob) uses for decryption as shown in Figure 3.1. Asymmetric algorithms have the advantage, that the key for encryption can be made public and only the key for decryption needs to be secure. Asymmetric algorithms are based on mathematical problems, hard to solve by current mathematical knowledge and computational power of computers.

So if a message needs to be secure in the next 50 years, the message has to be information theoretically secure (ITS). The common RSA encryption

is not information theoretically secure (ITS) and as soon someone manages to efficiently prime factorize, either by an algorithm or a quantum computer, all messages encrypted by RSA will not be secure any more.

In contrast to that, in symmetric encryption systems, Alice and Bob share the same key for encryption and decryption and as long as one can safely generate and deliver the key to his communication partner, the message is information theoretically secure (ITS). A sketch of the symmetric encryption is shown in Figure 3.3.

Symmetric algorithms are therefore the choice of encryption if the message needs to be safe in the future.

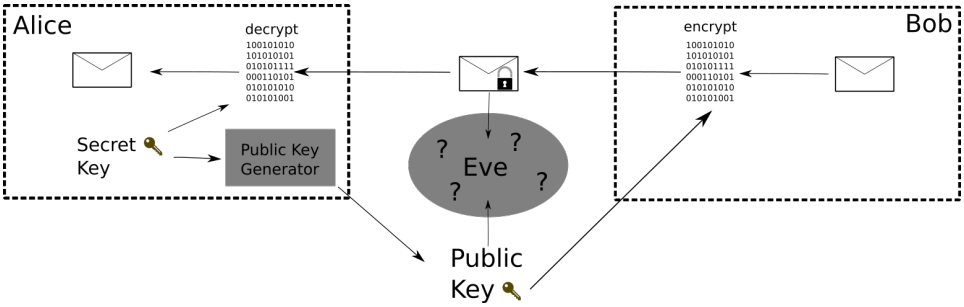

Figure 3.1: Example of asymmetric encryption: With the secret key, a public key is generated which is used for encryption. The Secret Key, which is only known to Alice, is needed for decryption.

To overcome that, the so-called one-time pad or Vernam cipher represents a solution. The one-time pad uses a key needs to be as long as the message to encrypt the message by a bitwise XOR operation.

As a result, the encrypted message is as random as the random key used for encryption. To decrypt the message, one has to do the same XOR operation on the message again as shown in Figure 3.2.

As long the random key generation is secure and the key is only used once, the one-time pad or Vernam cipher method is ITS [2].

However, the Vernam cipher is rarely used because of the fact that Alice has to share her secret key somehow with Bob, which has to be done by means of an insecure channel.

Classical encryption reaches its limit here. A new way to generate a totally random and ITS key for encryption is delivered by quantum mechanics, which will be explained in the next section.

3.2 Quantum Protocol

As no classical method of generating and delivering an ITS key is known, it has rarely been used. The one-time pad has to be transported by a trusted

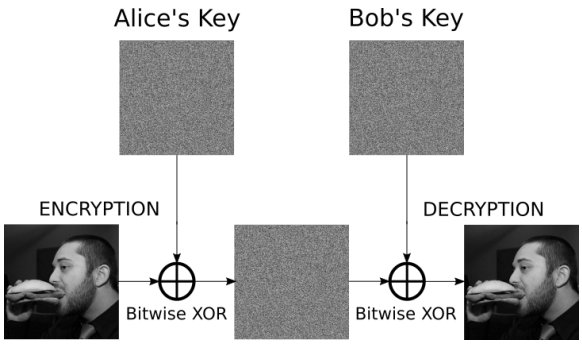

Figure 3.2: By applying a XOR operation on each bit, the message is as secure as the key. Do a XOR operation again with the same key, to get the original message.

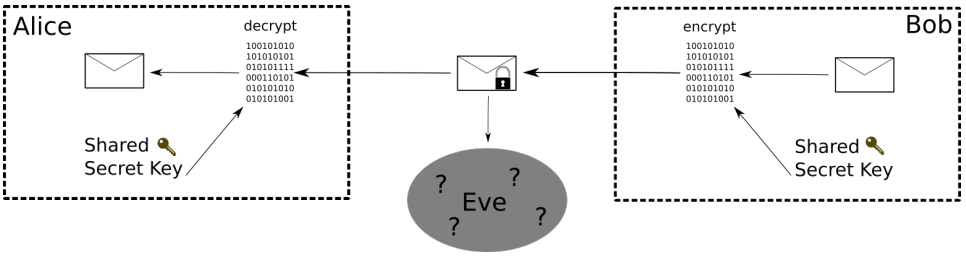

Figure 3.3: Example of one-time-pad symmetric encryption: The same key is used for encryption and decryption.

courier, which still contains a small percentage of uncertainty.

At the end of 20th century, people came up with a solution in the field of quantum mechanics. The method was first called quantum cryptography, but since a key is only generated by the laws of quantum mechanics and not manipulated, quantum key distribution (QKD) seemed to be a better name for the process. Quantum key distribution (QKD) solves the problem of generating and delivering an ITS key. It generates a key only known by the two members (Alice and Bob) which want to communicate. This key can be used in classical encryption (and decryption) and messages can be transmitted through classical channels like the internet. The basic idea of quantum key distribution is to exploit the property that quantum states can not be copied and that it is not possible to perform a measurement which leaves the state unchanged.

QKD encodes classical information with single quantum states (e.g. polarization of a photon), which still can be eavesdropped by a third party member (Eve). However, if Eve is listening during the key generation process, it can be easily detected by looking at the measurement result and therefore no valuable information is sent. Hence, Eve is able to drop the

transfer to zero in the worst case, but will not ever obtain relevant data. In reality, QKD protocols intend to differ in terms of quality, security and key generating rate. Error rates, noise and mechanical imperfection make these protocols much more complicated. Yet, eavesdropping can still be detected. Since the publishing of the first QKD protocol, numerous protocols have been proposed. This thesis focuses on BB84, which is the best known one and the basis for most of the other protocols, and entanglement, which was implemented for this thesis.

3.3 BB84 Protocol

The first protocol for quantum cryptography was proposed in 1984 by Charles H. Bennett, from IBM New-York, and Gilles Brassard from the University of Montreal; hence the name BB84 by which this protocol is recognized nowadays. They published their work during a conference in India totally unknown to physicists. We shall explain the BB84 protocol using the language of polarization, but clearly any 2-level quantum system would do so (e.g. spin) to encode it into bits as shown in Figure 3.4.

It is shown in Figure 3.5 and can be summarized as follows:

1. The transmitter (Alice) uses a photon source to bring photons randomly (and independently) into one of the four polarization states, |H>, |V>, |+> and |->, record them and sends the photons to the receiver (Bob).

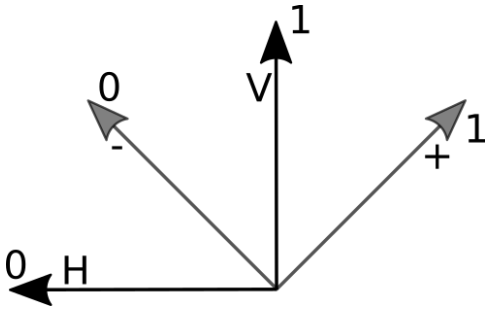

Figure 3.4: Quantum bit: Polarization representing their corresponding bit

2. Bob tries to measure the incoming photon with a randomly and independently to Alice's, base (|H>/|V> or |+>/|->). Both parties now share the so-called raw key.
3. Alice and Bob share their bases, which were used for generating and measuring photons. This data can be sent publicly, since there is no actual information in it. This step is called sifting and now both parties share the sifted key. Since Bob chose his measurement base randomly, 50 percent of all all measured photons are measured in the wrong

base/different to Alices' measurement. Therefore, 50 percent of the data will be discarded. Furthermore, if Eve would have managed to somehow intercept the communication, Eve could have detected photons. Eve could recreate the measured photons, which might somehow possible in the future. But by measuring the photons, Eve raised the error rate to 50 percent. Half of the data is already wrong. By doing an additional bell measurement(Alice or Bob), the error rate raises again by 50 percent. Invalid measured photons can not be measured in a valid way any more. Therefore, the overall error rate is significantly over 50 percent. Around 75 percent of all measured by Bob do not match with the result Alice get from the measurement.

4. In an ideal procedure, Alice and Bob would now share exactly the same key, but due to imperfection and various non-ideal effects, they do not. In a perfect system, 50 percent are dropped but in a real implementation is more between 40-48 percent, depending on the exact setup and hardware. Usually, all errors are treated as security violation.
5. The next step is the error correction process. To do that, a classical way is, to send parity of certain blocks of bits (e.g. Cascade). With the error estimation, Alice and Bob are able to tell how much information Eve might have received [13]. After error correction, Alice and Bob share perfectly correlated keys.

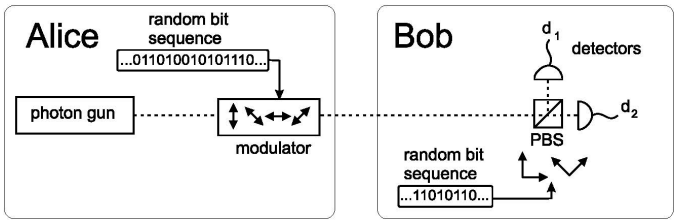

Figure 3.5: BB84: Due the fact all communication is not ideal, one can not distinguishes between noise and eavesdropping attacks. Since there is no way to reduce the noise to zero, eavesdropping attacks are always happening, and therefore we have to somehow use the "eavesdropped" key [2].

6. The next steps leads to the so called privacy amplification, in which Alice and Bob sacrifice a big part of the key, to gain security. There are numerous ways to carry out a privacy amplification. Also, privacy amplification methods are strongly developed further [3].

If Bob measures the photons' polarization at the correct base, he gets 100% the same bit value as the one Alice prepared. Also, if Bob chooses the wrong base, Bob still gets the correct bit value with a probability of 50%, which leads to the overall match of the bits of 75%. If Eve intercepts the communication, she can only measure the photon at random bases as shown in Figure 3.6. Therefore, the validity of the key is reduced to 75% before Bob

has the chance to measure it. Hence, Bob's measurement can only result in a maximum of a 67.5% match to Alice's bit values. Consequently, Alice and Bob know that Eve is listening and discard the faulty key.

During the quantum key distribution process, not all communication between Alice and Bob needs to be encrypted but authenticated. Bob has to make sure that he is talking with Alice, and not Eve. To do that, Alice and Bob have to share a secret key beforehand, to verify an ITS authentication. Because of that, quantum key distribution is sometimes called quantum secret growing. Especially new methods of privacy amplification methods are evolving constantly [3][21][2].

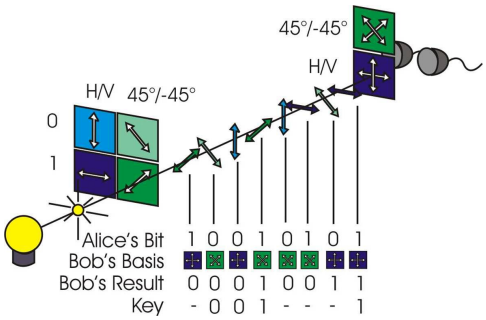

Figure 3.6: Alice prepares a random polarization and Bob measures in a random base(|H>/|V> or |+>/|->) [2].

12

3.4 Entanglement Protocol

As the BB84 protocol, which is the best known protocol, was introduced in the previous section, the entanglement protocol which is an important protocol in this thesis will get an overview next. QKD based on entanglement was first introduced by Arthur Ekert in 1991 [5]. Although the protocols differ in their experimental setup, the software part of the entanglement protocol is not different to BB84 and a few others.

A protocol for quantum cryptography using entangled states can be implemented by using two identical receiver stations (Alice and Bob) and a source for polarization-entangled states. It is shown in Figure 3.7 and can be summarized as follows:

1. Like in BB84, each station does a bell measurement and defines the polarization at a random base.
2. In contrast to BB84, Alice and Bob do not prepare and measure the same photon, but two polarized, entangled photons. Since, both share the same polarization, Alice and Bob receive the same raw key as a result.

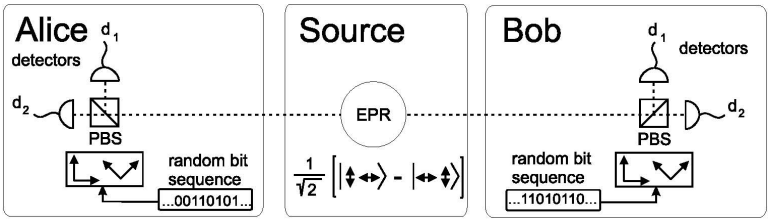

Figure 3.7: An independent source prepares an entangled photon pair and sends it trough a quantum channel (e.g. free space or fibre) to Alice and Bob [2].

3. Steps 3-6 are identically to BB84.

In Ekert's QKD protocol, the source of the photons is neither at Alice's nor at Bob's but in a third location. This represents a big practical advantage for future implementation. In BB84 Alice or Bob need a source if they want to communicate safely. A quantum source is the most expensive and complicated part of this setup. In reality, everyone who wants to do a QKD needs a photon source. In contrast to that, in entanglement, a single source can be used by multiple parties [2].

3.5 Summary

Classical encryption systems are based on mathematical assumptions and problems (e.g. RSA) which are hard to solve. No one can say for sure if this process can be eavesdropped already or if it can be in the near future. Quantum mechanics introduces a way to generate a ITS key and deliver it to the respective partners. Although most of QKD protocols are perfectly secure in theory, practical implementation still suffers from imperfectness of hardware. There is probably no way to overcome imperfectness of hardware, but with QKD, the level of security can rise and the insecurity shrinks to a negligible value. Common QKD protocols use a symmetric scheme where encryption and decryption are done with the same key by a simple bitwise XOR operation. In quantum protocols, properties of light can be decoded in bits. A very common method to do so is to use the polarization of light as respective values. With a bell measurement, polarization can be detected and digitalized. The BB84 and entanglement protocols differ in their experimental setup, but share the same process after the creation of the coincidence table.

Chapter 4

Synchronization

In both quantum protocols described, two stations (Alice and Bob) exist. Both could either be on a single machine, so they use the same clock to detect an incoming photon, or sit on different stations. These can be one metre or even 143 km apart, like in the experiment done by the Institute of Quantum Optics and Information from Tenerife to La Palma [11]. To identify the entangled photon pair at Alices' and Bobs', one needs to synchronize the clocks Alice and Bob are using. Photons are small and fast. To detect them, highly sensitive detectors are needed. Also, hardware is needed, which is capable of identifying the exact time at which the photon arrives. In this chapter, clocks and their typical errors are explained (see section 4.1 as well as how coincidences can be found on a single machine, how synchronization of two distant machines can be realized and how one finds coincidences there.

4.1 Clock Accuracy and Stability

A typical clock is generated out of a crystal oscillator, which is neither produced perfectly nor do share crystal oscillators their imperfections. Two of an oscillators clock's main properties which are also important for this thesis are accuracy and stability. The clock accuracy describes how well the actual frequency matches the specified frequency. Clock accuracy might be affected by factors such as the quality of the oscillator crystal and how the oscillator was assembled. The absolute minimum and maximum frequency of a clock at 10MHz and with an accuracy of 0.01% can be calculated as follows:

$$\begin{aligned} MaximumFrequency &= 10,000,000 + (10,000,000 * 0.0001) = 20,001,000 \\ MinimumFrequency &= 10,000,000 - (10,000,000 * 0.0001) = 19,999,000 \end{aligned} \tag{4.1}$$

The frequency diversity is located somewhere between minima to max-

ima. Therefore, compared single moments of two clocks always differ in their accuracy. However, another point is the stability of a clock. Accuracy is the difference between single clock pulses, but what about longer time spans?

The stability specification provides a measure of how much the frequency varies over time and how well the oscillator frequency resists fluctuations. The dominant factors that affect stability are a variation in temperature, ageing over time, supply voltage, shock, vibration, and capacitive load that the clock must drive. Two free running clocks eventually drift away from each other. Drifting clocks show different dates over time. After one day, the second clock might be a few seconds ahead of the other one. This effect is called stability. An unstable and inaccurate clock would drift away from each other in a long term and even in a short-term measurements; it does not have a continuously result. However, a perfectly accurate and stable clock would produce the desired specifications. Still current technologies are not capable of delivering an almost perfectly stable and accurate clock at the same time. The more stable a clock gets, the more its accuracy decreases as shown in Figure 4.1 .

To be able to detect coincidences or arriving photons, a clock with a very high resolution, high stability and high accuracy is needed. At present, typical, commercially available resolution is between 1ns and 100ns [16][25][26].

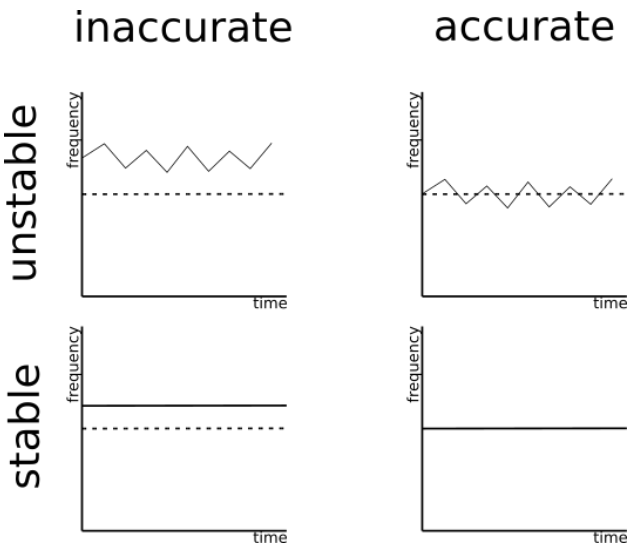

Figure 4.1: The dotted line represents the specified, desired frequency, and the solid line represents the actual frequency generated by the oscillator [25].

4.2 Coincidences

If Alice and Bob want to perform QKD, coincidences have to be found. Coincidences in quantum mechanics are the detection of entangled photons at the time-tagging module. To find these coincidences over the distance, an offset between Alice's and Bob's clocks has to be known. To identify this offset between clock A and B consider two distant parties. Alice and Bob are preforming quantum experiment with entangled photons. Each of them holds a clock, namely clock A and clock B. We assume the two clocks run at the same frequency for the first step. In order to identify the photon pairs, the two parties have to know the time offset between clock A and B. This can be done with purely quantum signal. An entangled photon pair emitted by a source and send to Alice and Bob will be detected at time t^a and t^b. Additionally the travel time difference t^d has to be considered. Therefore, the time offset can be expressed as follows [20]:

$$\Delta t = t^a - t^b - t^d$$

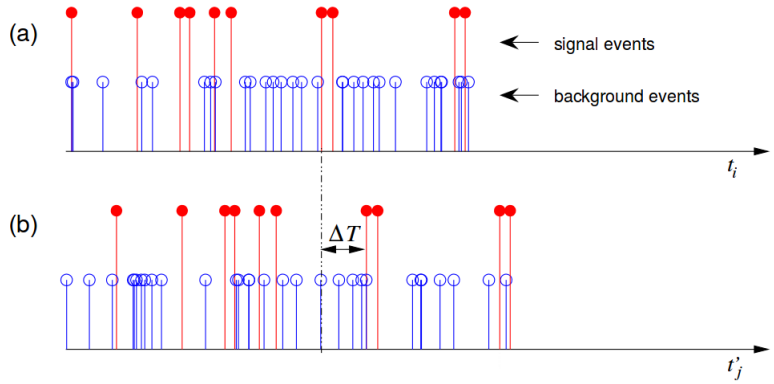

Figure 4.2: Effect of time offset and clock drift on photo event sets. Trace (a) represents the event set on side A, trace (b) an event set on side B with a time offset ΔT, but the same reference clock frequency [9].

Also, the relative frequency difference between the two clocks might drift away. This happens due to the lack of accuracy and imperfections of the clock. Therefore, both time traces differ from each other and a certain time window is chosen to actually detect the coincidences. To do that, trace (b) will be shifted along trace (b). When the exact ΔT time is matched, a peak coincidence value will appear like in Figure 4.2.

4.3 Hardware Synchronization

To be able to find entangled photon pairs, one has to detect photons and their polarization and compare them to each other. For this purpose, a photon counter is normally used. A Photon counter consists of 4 input ports, which are capable of low voltage differential signaling FLVDS) detection [10] and represent polarization states of |H>, |V>, |+> and |->. For this thesis, avalanche diodes are used for detection, although there are numerous other detection methods, like photo multipliers or superconducting transition edge bolometer. Still, single-photon avalanche photo diodes (SPAD's or APD's) are the most common detectors in optical application, since they have a quite good cost-benefit ratio. The SPAD is operated in Geiger mode. A photon triggers the SPAD and a LVDS input of an electronic device (e.g. FPGA) detects the event [8].

To detect an entangled photon pair, one has to compare Alice's and Bob's detections by using a simple AND operation. This can be seen in Figure 4.3. This is possible due to the very short time delay between the entangled photons using the same hardware. Alice and Bob share the same location. This simple process is called coincidence detection. Due to imperfection in hardware, one has to consider the jitter of the detectors in the order of 0...5ns. To compensate that, a coincidence time window has to be specified during which the "earlier" detection waits for a certain amount of time. The larger the time window is, the higher the probability that two detections are erroneously recognized as entangled photon pairs. These are not called coincidences but accidentals. The more accidentals occur, the more the error rate in QKD rises [2].

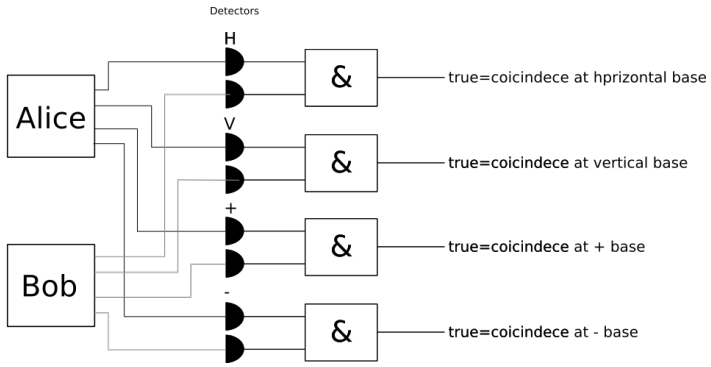

Figure 4.3: Coincidence finding at a single location by doing a simple AND operation

4.4 Software Synchronization

As described in the previous section, it is quite easy to find coincidence on a single machine. Alice and Bob use the same clock for their measurement, thus they have the same time base. A problem occurs if Bob moves to another location. Since Bob is not on the same machine any more, Bob uses a different clock for his time base. Due to mechanical imperfection, this other clock can not be as accurate and stable as the clock Alice is using. Also, as the distance rises and Alice and Bob are placed on different speed or gravitational bases, effects of time dilation might occur, which lets clock tick at a different pace. In reality, such effects have to be considered if a source is installed on a satellite. To compensate these effects, clocks have to be synchronized as well as possible (e.g. GPS), and further corrected with mathematical formulas [9]. After the synchronization of the clocks is done, correct timestamps of photons can be logged. These logged timestamps can be shifted to each other to find the coincidence peak.

A coincidence searching diagram is shown in Figure 4.4 and program can be implemented as follows:

- Find the first timestamp in Alice's file.
- Find the first timestamp in Bob's file.
- Calculate the offset between Alices' and Bobs' clock.
- Shift Bobs' timestamps along Alices' until Alice and Bob share the same time window.
- If Alice and Bob share a coincidence in the same time window, add it to the coincidence table.
- Redo these steps with all timestamps.

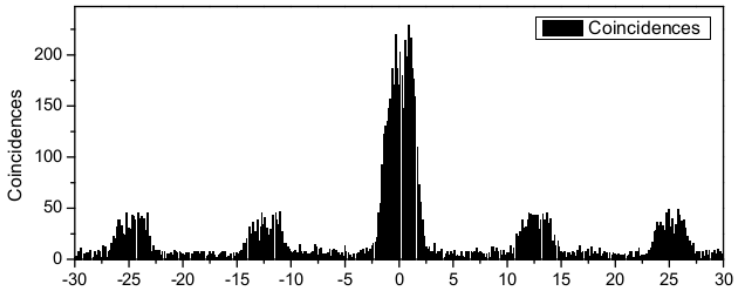

Figure 4.4: Time window shifting: when the correct offset is found, a coincidence peak will appear [19].

For this task, the quantum key distribution software of the Austrian Institute of Technology is used. This software is capable of finding coincidences and creating keys for encryption as shown in Figure 4.5.

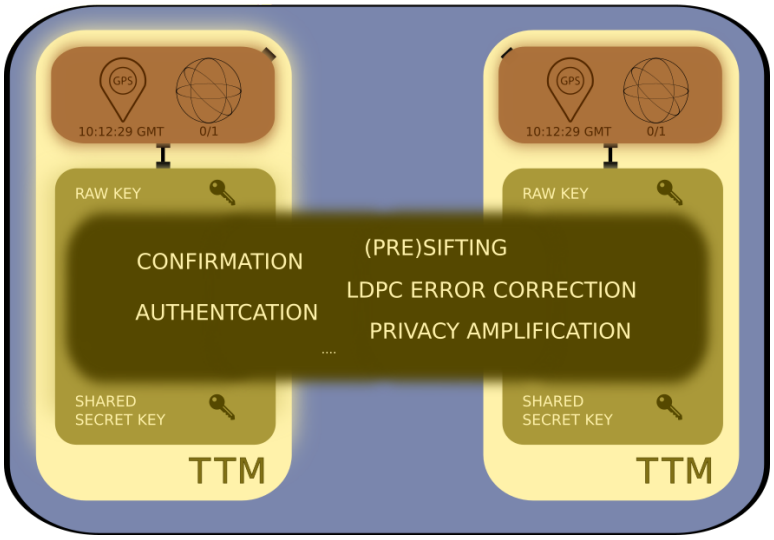

Figure 4.5: TTM: Time Tagging Module, hard clock synchronizing per PPS GPS, measuring polarization, then QKD takes care of further synchronization

4.5 Summary

Photons are very small, and very fast. To detect something that small, very good clocks are needed. The higher the resolution, the better the coincidence rate and the faster a key can be generated. In quantum protocols the key has to be as long as the encrypted message. Therefore, the key generation rate limits the bit rate for transferring data. The bit rate improves as the quality of the clock rises. Accuracy and stability are very important for such a clock. To gain a key out of a quantum protocol, coincidences have to be detected by different modules run by two different clocks. Therefore, the clocks have to be synchronized. Long distance synchronization can be achieved by using GPS powered, 10 MHz signals. Still, clocks are drifting away from each other and errors occur. The remaining errors can be corrected by software though.

Chapter 5

Quantum Key Encryption

5.1 Motivation

Encrypting messages is not new. Even in informations technology encryption has been done since almost the beginning. Early encryption systems were, compared to today, pretty simple and easy to crack. But with the rising of computer and the internet a much more complex system has been evolved. A very common encryption nowadays is the RSA-encryption. This encryption system, like a lot of other currently used systems, are secure because of mathematical assumptions and missing efficient algorithms for classical computers. Hence, if someone would develop such an algorithm or existing quantum algorithms are implemented, current encryption systems would be as easy to crack, as encryption system from the early 21th century are for us now.
Quantum key distribution offers a method of creating and deliver a key to two parties in a secure a way. In difference to classical encryption methods it is not based on mathematical algorithms but the law of physics.

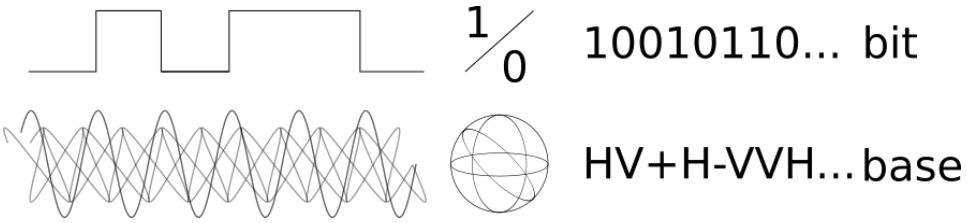

Figure 5.1: Comparison classical vs. quantum bits. Quantum bits in in QKD need measurement data from the second party to produce valuable information.

A classical bit can be measured, copied and forwarded and there is no possibility one can distinguished between the original bit or the copied bit. They are 100 percent identical. In contrast to that, a quantum bit has a

completely different concept. A quantum bit, represented by the polarisation of a photon, can neither be measured completely nor copied due the law of physics. Additionally, even if a third party would somehow manage to find a way to copy a quantum bit, the final result have to be compared with the second party first, to get valuable information. This is sketched in Figure 5.1.

This properties of QKD is reason enough to do further research and bring quantum cryptography to a usable level.
Quantum key distribution exists since 1984 and a lot has happened since then. The first implementations were pretty simple compared to today. Moore's law is going on and detection of single photon detection is as efficient as never before. Because of that a lot of processes can be simplified and automated.

Until now, the measurement of the photons and generating of a quantum encryption were separated. A hardware module was responsible for recognizing and logging the events to a file. This file then were transferred to a pc and then the key was generated by parsing the file and using programs like Mathematica or Labview. To simplify this process, and make it more efficient, a System-on-Chip modules is used to develop a module, capable of detecting quantum events and generate a usable key for encryption. To test this, a QKD system was provided by IQOQI and adapted for an implementation of an entanglement QKD protocol.

5.2 Overview

Quantum key distribution (QKD) provides a modern method of sending a more secure messages to a trusted receiver. Also, the probability that the message stays secure during its lifetime also rises significantly. QKD has been implemented a couple of times already but the abstraction layer is still low. Although commercial applications still fail to be practical, technology levels rise pretty fast and further abstraction layers can be made on stable assumptions. In the experimental setup shown in 5.2, a new layer was made where time-tagging of quantum events and quantum key growing happen on the same machine, on an embedded device.

1. The source produces two polarized, entangled photons by using a laser and a tempered crystal for single photon down conversion(SPDC).
2. The photons are send to Alice and Bob. This can be done by fibres or free space.
3. The arriving photon is measured using a bell measurement.
4. The photon are detected by avalanche photo diodes which produces a electrical signal. This signal can be detected by an input pin of the SoC. The arriving time of each photon is logged.

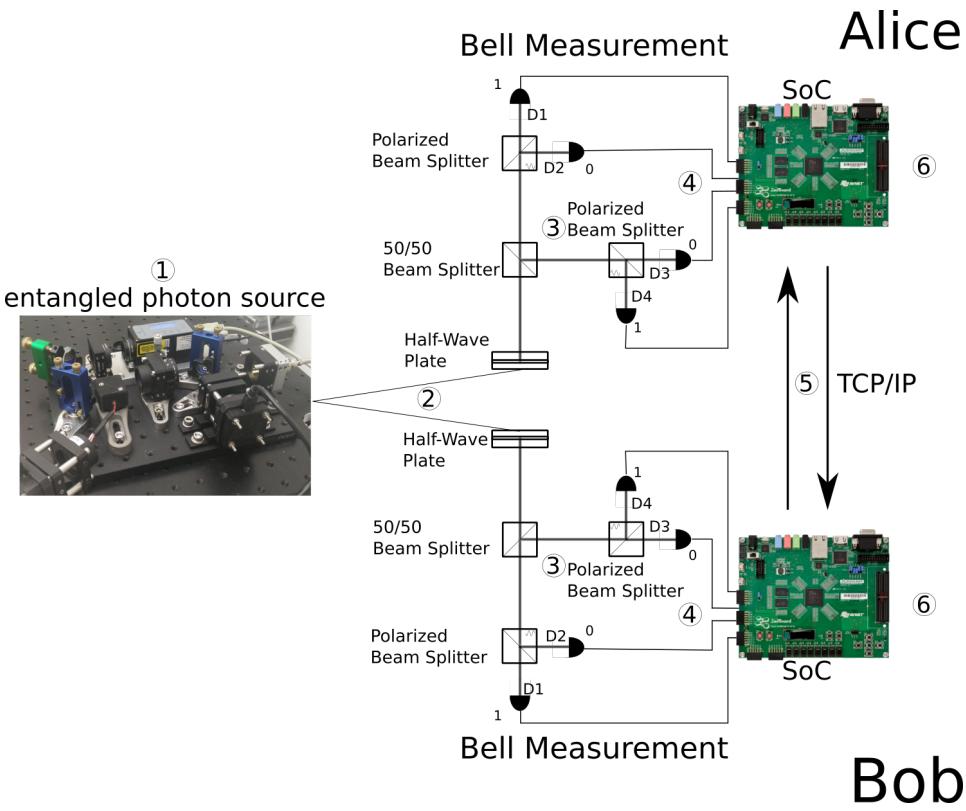

Figure 5.2: Overview of the setup.

5. Alice and Bob share their respective events and bases which were used in the measurement.
6. After sharing, software for error correction is started.
7. After the error correction process Alice and Bob share identical keys.

5.3 Scenarios

In contrast to the BB84 protocol, in the entanglement protocol as shown, Alice or Bob do not need a source, the most complex part, on their own. The source can be provided by a third party which gives a practical advantage for commercial use, since one source can be used for a wide range of Alices and Bobs. This can either be done on the ground, like it has been done from Las Palmas to Tenerife [11] or, which is more likely, using a satellite as distribution system as shown in Figure 5.3. For the module it is irrelevant if the key distribution system is at the laboratory, on the ground or even in space. Beside likely higher error rate the key generation process is always the same.

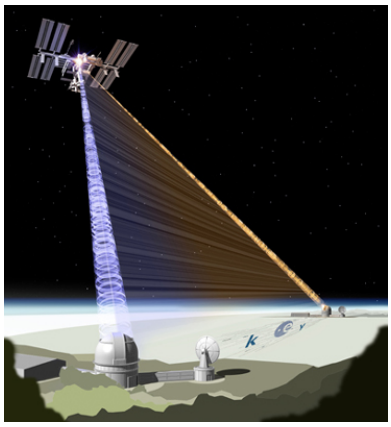

Figure 5.3: QKD using a satellite to distribute to Alice and Bob [24].

5.4 The Source

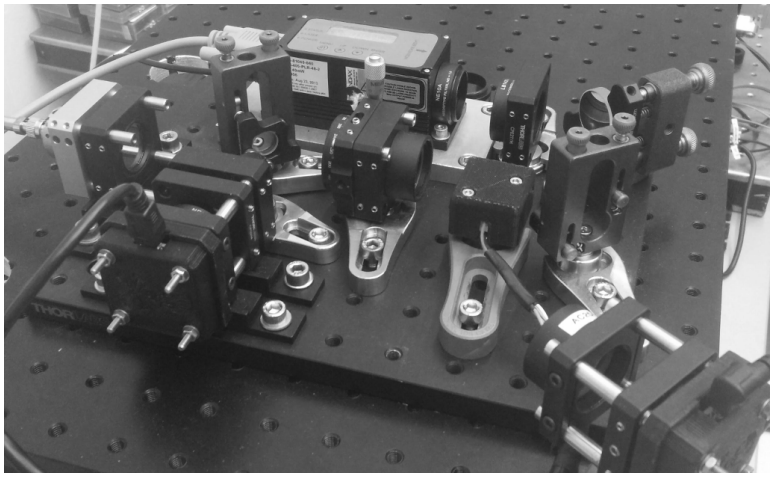

Figure 5.4: Entangled photon source.

The source, provided by the Institute for Quantum Optics and Quantum Communication (IQOQI), consists of a single-pulsed laser, emitting 405nm (ultraviolet) photons with random polarization. A picture of the source is shown in Figure 5.4. A density filter prevents overloading of the time-tagging module. The photon is converted from 405nm to two 810nm (red) polarized, entangled photons by a heated crystal. Because of asymmetrical behaviour of the crystals, |H> and |V> polarized photons do not travel at the same speed through the crystal. To compensate that a filter is used which delays the photons the other way around, to make both photons indistinguishable from each other. To deliver the photons to their respective partner, mirrors

are used to route the path.

5.5 Bell Measurement

The two entangled photons are sent through quantum channels to the receivers Alice and Bob. Current practical quantum channels are fibres and the free space. Alice (A) and Bob (B) use two identical hardware modules

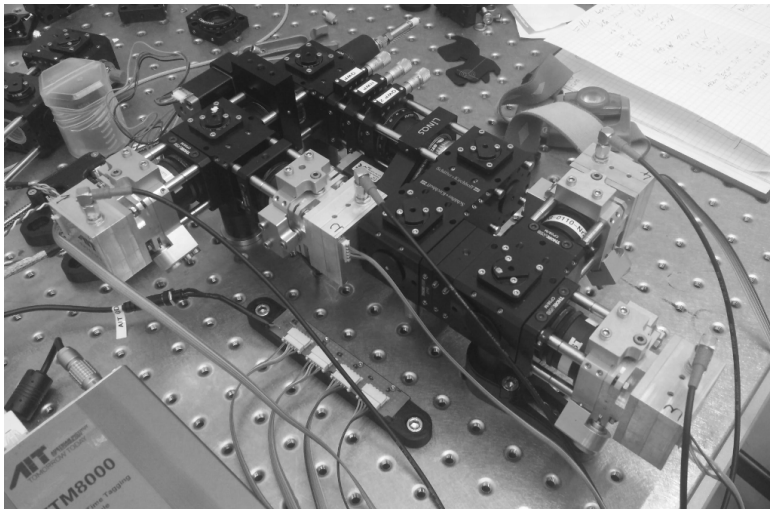

Figure 5.5: Bell measurement.

consisting of a bell measurement and a time-tagging module (TTM). Using a beam splitter the photons are randomly split up to measure the polarization on a single photon in either |H>/|V> or |+>/|-> bases. Each base measurement is done by a polarized beam splitter and two avalanche photo diodes(APD). A picture of the bell measurement is shown in Figure 5.5.

5.6 The System-on-Chip

Quantum events triggered by photo diodes are captured and logged by the time-tagging module (TTM). The FPGA on the SoC is configured as time-tagging module with a time resolution of 78.125ns and triggered by LVDS signals of the APDs. To hit the same time frame on Alice's and Bob's module, the clocks are synchronized by a GPS clock running the local clock with an 1pps signal (10MHz). This minimizes the error rate significantly since the standard clocks on the board are not stable enough for proper quantum key distribution. As soon as the clocks are synchronized, Alice and Bob are capable of detecting coincidences between their and their partners's module.

On current SoC, CPU cores can be used to boot a Linux distribution to run quantum key encryption software which is shown in Figure 5.6. This software is necessary for coincidence finding and key creation. To do this, Alice and Bob communicate over TCP/IP and compare their results by sharing their timestamps, which they got from detecting quantum events. Due to the fact that Alice and Bob rarely share the exact same distance and channel type, a delay for calibration has to be determined before each run.

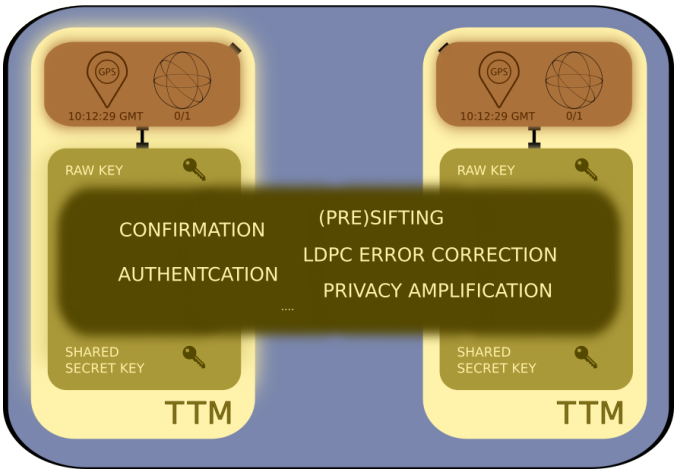

Figure 5.6: TTM: Time Tagging Module, hard clock synchronizing per PPS GPS, measuring polarization, then QKD takes care of further synchronization

After that, Alice and Bob share their coincidence table. This table is used for creating a information theoretically secure key. In the next stage, Alice and Bob use classical channels to continue with the key creation. Alice and Bob go through a couple of stages to determine if someone was listening during a previous process of creating the coincidence table. The table is sifted, error corrected, confirmed and privacy amplificated to make sure, that Alice was talking with Bob and not with someone else. However, if the error rate is too high, the possibility that Eve was listening rises and the key is discarded. Otherwise, privacy amplification gets rid of a big part of the created key to gain security. This procedure can be done several times to gain even more security. Although QKD is secure in theory, actual implementation has drawbacks concerning imperfection of the source, detectors and transmitting channels which adds insecurity into the key. Again, all transmitting errors are treated as eavesdropping.

As soon this is finished, Alice and Bob share the same ITS key and are capable of sending an encrypted message to each other. Each key is used only once, to remain secure. Yet, since the rate of creating such a key is not as high as data rates in use nowadays, the key created can be stored on Alice's and

Bob's module for future use. If one considers that future implementation will be most likely done via satellite, keys can only be created if the satellite is in range and at the right position. Also, current methods share the property that they only can efficiently applied at night, which further supports the feature of saving the created keys. Of course, the keys are only as secure as the place, where the keys are stored. The overall process i shown in Figure 5.7.

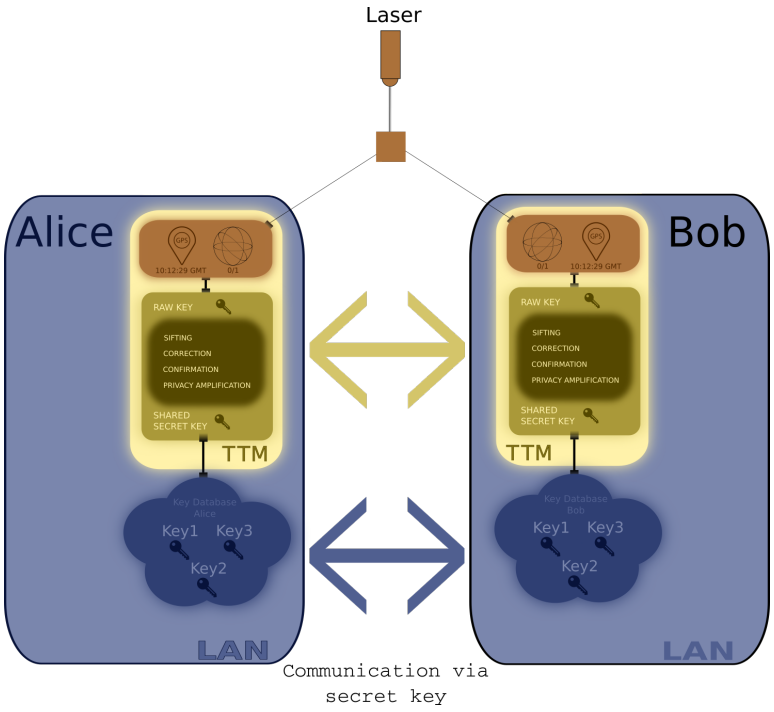

Figure 5.7: The full QKD system: A source emitting 2 polarized, entangled photons is measured by Alice and Bob which create a key for encryption.

Chapter 6

Hardware

To actually do QKD, and go beyond theoretical assumptions, hardware is needed which is capable of creating, measuring and processing a quantum key. Although there is still a long way to go, QKD does not require a quantum computer. The actual quantum part is done with light, to be more precise photons. Photons can be much more easily manipulated and detected than any other quantum. Very precise lasers can be bought in various operating modes, colours and rates. For manipulating photons. Optical devices are perfect for manipulating photons. Optical lenses, crystals and filters are constantly developed further and are that efficient at quantum level as we are used to them. Photon detectors are available in various shapes and it's quality and performance is determined only by the price one intends to pay for it. As for processing, a standard CPU and operating systems can be used. In this thesis, detecting and post processing is done on a single machine, and not spread on various devices. This allows to go a step further in terms if abstraction concerning quantum key distribution.

6.1 Zedboard

The ZedBoard is a development board for the Xilinx Zynq-7000 All Programmable System-on-Chip (SoC). The board consists of a FPGA, where the time-tagging module is configured and 2 ARM cores clocked with 900 MHz run a custom made Linux. The Linux handles common communication protocols and file management. The operating system is also capable of running the latest Quantum Key Distribution (QKD), developed by the Austrian Institute of Technology (AIT). Additionally, several expansion connectors are included as well as VGA, Ethernet, HDMI, Led Display, Serial, and LEDs [23]. The available peripherals are shown in figure 6.1.

Features:

- Zynq-7000 All Programmable SoC XC7Z020-CLG484-1
- Memory

1. 512 MB DDR3
2. 256 Mb Quad-SPI Flash
3. 4 GB SD card

- 10/100/1000 Ethernet
- PS and PL I/O expansion (FMC, Pmod Compatible, XADC)
- Linux/Android/RTOS development
- Embedded ARM processing

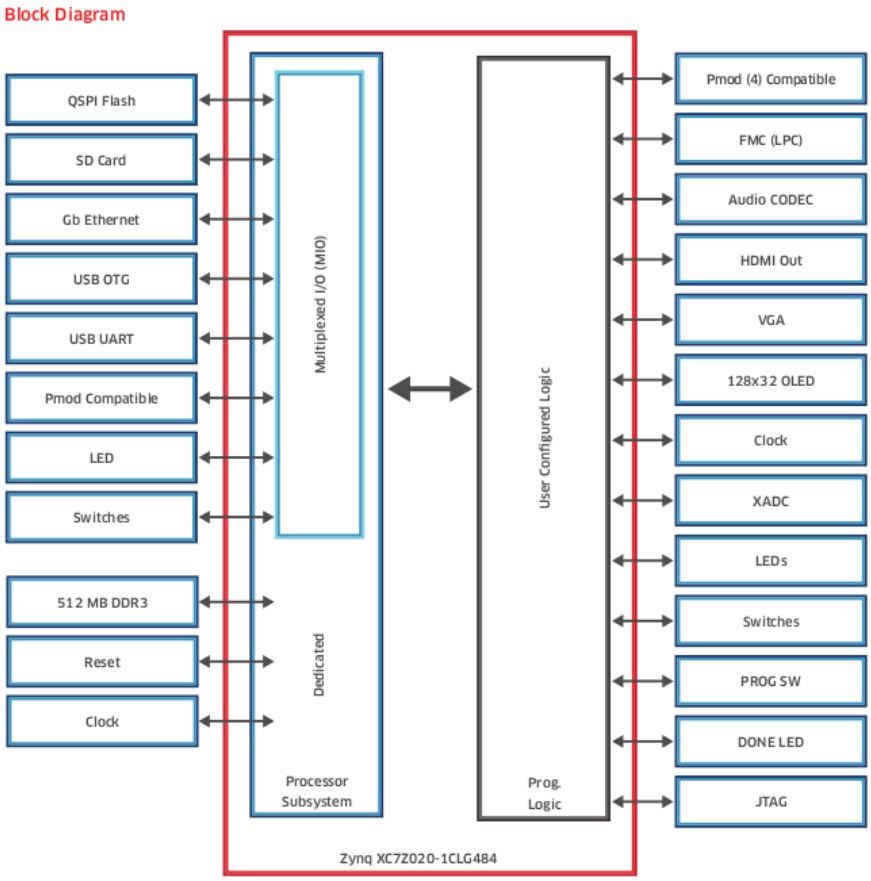

Figure 6.1: Zedboard Zynq-7000 features [23].

The functionality of the QKD is not limited to for this specific board, any SoC (FPGA+CPU) might work too. An FPGA is implicitly necessary to implement the time-tagging module ans a CPU for actually run an operating system to heavily increase the flexibility for processing key. For the implementation, an Ethernet connection is used, but of course, any other

connection type can be used as well. Ethernet, however, has the advantage of data rate and long distance capability.

6.2 Time-Tagging Module

A Time Tagging Module (TTM) is needed to resolve a quantum event of a detected photon in one of the polarization bases. As this has to be done as fast as possible, software processing is far too slow. Therefore the time-tagging module is configured on the FPGA-Part of the SoC and provides a driver (shared library) for any operating system. The configuration were provided by IQOQI-Vienna. On Linux, the TTM appears as device file at the operating system and has multiple C functions for configuration and usage. The most important functions are declared as follows:

Listing 6.1: ttinterface.h: Header to access the TTM.

```
/*
 * Switch the mask of the LVDS input.
 */
void SwitchLvds(int mask);

/*
 * Calibrates all input channels uniform delay.
 * Returns 0 if no errors appeared
 */
int Calibrate(void);

/*
 * This function reads the next tag from the FIFO.
 * channel: 1-8
 * time: time in 78,125 ps units
 * Returns 1 when tag is present, 0 otherwise
 */
int ReadNextTag (int * channel, long long * time);

/*
 * Use 10MHz external clock.
 */
void Use10MHz(int use);
```

A external 10 MHz clock can be used for a much better clock stability. LVDS signals can be detected and resolved every 78.125picoseconds.

6.3 GPS Module

In this experiment, to roughly synchronize Alice's and Bob's clocks, a GPS module is used. LC-XO-PLUS GPSDO emits a 10 MHz reference clock and delivers 1 pulse per second (PPS), which means that the clocks are synchronized every second and that they are as accurate as 1 nanosecond, which is

enough accuracy to archive a decent result. It may acquire up to 50 GPS signals, a low-noise 5V CMOS 10MHz and 3.3V CMOS 1PPS output, USB control interface, precision voltage references, and digital-to-analogue converter (DAC) A picture of the module is shown in Figure 6.2. It generates a low-noise, 10MHz signal with a typically better than 1E-010 precision and accuracy (0.1ppb), better than 250fs jitter (20Hz to 2MHz) and a 1PPS signal with typically better than +/-25ns stability when locked to GPS. This module is used for running the clock on Alice's and Bob's module as stable as possible. It may be operated from a 5V power supply and a GPS antenna with a length of 10m [15].

Figure 6.2: gps clock: 1pps

Chapter 7

Software

7.1 Operating System

Using a custom Linux on the SoC implies having two options: either useing the Petalinux framework provided by Xilinx or creating a custom made Linux distribution by using Yocto. The Yocto Project is an open source collaboration project that provides templates, tools and methods to help you to create a custom Linux-based system for embedded products, regardless of the hardware architecture. Owing to the fact, that the Petalinux distribution is based on the Yocto Project, hence bitbaking (building a distribution) is far more flexible for future references, the Yocto Project is used. By creating a Yocto layer the whole distribution, including all key processing tools, can be easily adapted to multiple other architectures, including ARM, PPC, MIPS, x86, and x86-64. The Yocto Project is an open source and has a strong backup from many hardware manufactures, open-source operating systems vendors and electronic companies.

A new meta-layer was created based on the openembedded-core layer: meta-qkd. meta-qkd includes everything the QKD software of the Austrian Institute of Technology, it is discussed in the next section, needs for compiling and running on the embedded system.
To compile a custom Linux including QKD, the following can be done:

- get the source code

```
1 $ git clone git://git.openembedded.org/openembedded-core
2 $ cd openembedded-core
3 $ git clone git://git.openembedded.org/bitbake
4 $ git clone https://github.com/Xilinx/meta-xilinx.git
5 $ git clone https://github.com/mordeng/meta-qkd.git
```

- source the paths

```
1 $ source source oe-init-build-env
```

- add in config\local.conf

```
1 MACHINE ??= "zedboard-zynq7"
```

- add in config\bblayers.conf

```
1 BBLAYERS ?= " \
2   openembedded-core/meta \
3   openembedded-core/meta-xilinx\
4   openembedded-core/meta-qkd\
5   "
```

```
1 IMAGE_INSTALL_append = " aitqkd-source"
2 IMAGE_INSTALL_append = " ttm-driver"
3 IMAGE_INSTALL_append = " ttm-lib"
4 IMAGE_INSTALL_append = " ttm-keystreaming"
5 IMAGE_INSTALL_append = " ttm-demo"
```

- compile the minimal image

```
1  $ bitbake core-image-minimal
```

- run the image in an qemu emulator

```
1  $ runqemu qemuzynq nographic
```

Yocto bitbaked (compile) its own custom made Linux distribution for quantum key distribution for the Zedboard. This may take a while. After this process is finished a full Linux image can be found for booting the Zedboard with QKD abilities [31]. This embedded Linux distribution created, has full support to any peripherals on the Zedboard and is able to use the full advantage of using the FPGA on the SoC. Although, very few of the peripherals available are used, future implementation will profit. Furthermore, the whole Linux distribution can be easily ported to any other common SoC on the market by simply change a few variables.

7.2 Austrian Institute of Technology Quantum Key Distribution

Quantum cryptography is rooted in fundamental quantum mechanical effects, such as the famous EPR effect (1935 - Einstein, Podolsky, Rosen at the University of Princeton, USA). These effects enable the simultaneous generation of two identical, absolutely random bit-sequences in two distinct locations which are principally inaccessible to a potential eavesdropper. These sequences can be employed as a key for data encryption and thus provide the foundation for secure communication. Quantum cryptography has been the object of intense research in the last two decades. The research results convincingly demonstrate that practically secure communication based on quantum cryptography is generally speaking feasible and within reach [29]. In 2004 the Austrian Institute of Technology (AIT) has demonstrated the

first successful bank transfer using QKD. The AIT is developing a QKD software which is capable of performing actions like coincidence finding and key generation. The Software is open-source since 2010 and has been redeveloped a couple of times since then. At the time of writing, the QKD software is at release 10 and can be viewed here [29]. The Yocto recipe created consists of the following file:

- Line 1-4: Meta-data of the recipe.
- Line 5-6: Libraries needed to compile the software.
- Line 9-10: Checksum of the source code.
- Line 12-15: URI to source code and patches.
- Line 17: Additional programs needed during runtime.
- Line 19: Inherit from recipe classes of the Yocto Project for compiling.

Listing 7.1: Yocto recipe for AIT QKD software

```
1 DESCRIPTION = "AIT QKD R10 Software"
2 SECTION = "examples"
3 LICENSE = "CLOSED"
4 PR = "r0"
5 DEPENDS = "libtool dbus boost"
6 S = "${WORKDIR}/git"
7
8
9 SRC_URI[md5sum] = "5ffeb2b528a0f7411b3f273fc5a93771"
10 SRC_URI[sha256sum] = "
       a214bc1e94ffad4e8fa22de4994083cf29a5c21a17b0f3f031813190399a24a1"
11
12 SRC_URI = "git://git-service.ait.ac.at/quantum-cryptography/qkd.git;
       protocol=https;rev=v9.9999.4"
13 SRC_URI += "file://cmakelist_compiler_settings.patch;patch=1"
14 SRC_URI += "file://cmakelist_remove_latex.patch;patch=2"
15 SRC_URI += "file://cmakelist_install_tests.patch;patch=3"
16
17 RDEPENDS_${PN} += " zeromq qwt libicui18n"
18
19 inherit qt4x11 pkgconfig cmake
```

Also, the software is still under heavy development, therefore, the implementation might also be adapted to future changes. This is possible with the Yocto Project. A recipe was created to download, compile and install the AIT QKD R10 software. Since changes are committed to the GIT repository almost weekly, a specific revision was chosen. Furthermore, small patches were needed to successfully install the QKD software. The patches include small changes like more tolerant compiler flags and removing necessary libraries for latex to compile the handbook, which is not needed on the embedded device.

The purpose of this software is, to have a fast and simple application

which processes any needed step from a quantum event to a shared key between Alice and Bob. The software has to be multi-threaded and to run on exotic architectures too, to be flexible in the future. Furthermore, the software has to be built up in a way in which on can exchange single modules, since there is further development in different stages. Privacy amplification is one of these process steps, which will be replaced for sure in the future.

The AIT QKD R10 Software graphic shown in Figure 7.1 can be summarized as follows:

1. Quantum Event Acquisition: The AIT gets hold of a coincidence table created out of the time tags of the TTM module to create the raw key.
2. Sifting: This stage converts the raw key to the sifted key.
3. : Error Correction: The errors correction , created by imperfect measurement devices or eavesdropping, runs here.
4. Confirmation: Additional comparison of the key is done, due further possible errors based on stochastic processes.
5. Privacy Amplification: During step 3 or 4 some information might be leaked to third parties. Therefore, parts of the key are dropped to gain security.
6. Key Storing: After the creation, the key is stored for further use in encryption and decryption.

The remaining stages seven to thirteen in Figure 7.1 show further processes which is done over the network [12].

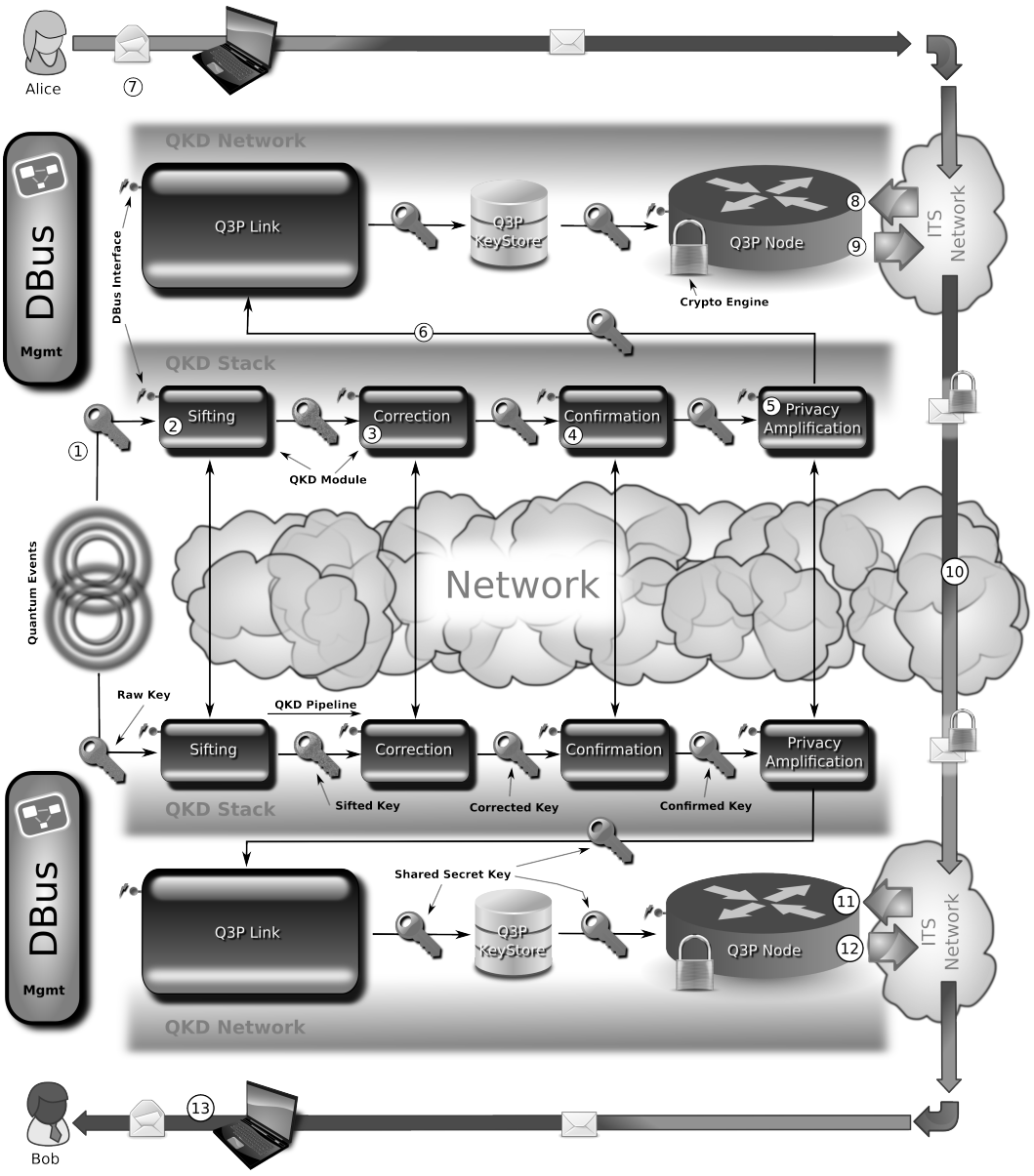

Figure 7.1: AIT QKD software architecture [29].

Chapter 8

Verification

8.1 Motivation

The embedded module developed in this thesis may detect this flaws and reconstruct the faulty key using the AIT QKD software, although some information is lost. This module is capable to measure and create a raw key and post process all necessary steps to gain a usable key for secure communication. The module uses the QKD software of the AIT to generate a quantum key for encryption out of the entanglement setup described in chapter 8.1.

8.2 Commissioning

8.2.1 Changes

Due to the time constrains, the original setup had to undergo some changes. The final time-tagging module, which should have been delivered early, was heavily delayed. This has shrunk the effectivity of the setup but the AIT QKD software can still be tested. Instead of two modules representing Alice and Bob, a single module is used to simulate both. Also, the bell measurement has been modified to make QKD possible. Instead of two times four detectors, only four detectors are used. The beam-splitter, which decides on the base, was omitted and therefore the base is static. In this case, only coincidences in one base (|H>/|V> or |+>/|->) can be detected. This reduces the coincidence rate significantly but the protocols and software functionality remain the same. Additionally, actual synchronization via GPS is also obsolete, since Alice and Bob are now driven by the same clock.

8.2.2 Source

The source is powered by an 405 nm laser which has to be switched on and the crystal for the SPDC conversion is tempered depending on to environ-

ment. Other parts of the source are static and optical. Therefore, they do not have to be switched on separately. The light is then delivered to the bell measurement by using fibres, this ensure a low loss rate.

8.2.3 Bell Measurement

Also the APDs have to be temperature controlled with a fan, so they do not overheat. Everything else is again optical and do not need further activation.

8.2.4 System-on-Chip

The Zedboard is using an SD card to configure the FPGA and load the Linux distribution. After successful boot of the Linux a service is started to control the status of the QKD program. The program can be started over TCP/IP or a serial connection.

8.3 Linux Distribution

The Linux distribution successfully boots up and loads necessary drivers to get access to the Ethernet connection and the TTM on the FPGA. This is shown on Figure 8.1.

```
U-Boot 2014.01-dirty (Nov 06 2014 - 20:10:36)

I2C:   ready
Memory: ECC disabled
DRAM:  512 MiB
MMC:   zynq_sdhci: 0
SF: Detected S25FL128S_64K with page size 256 Bytes, erase size 64 KiB, total 16 MiB
In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
Hit any key to stop autoboot:  0
Device: zynq_sdhci
Manufacturer ID: 27
OEM: 5048
Name: SD04G
Tran Speed: 50000000
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 3.7 GiB
Bus Width: 4-bit
reading uEnv.txt
249 bytes read in 7 ms (34.2 KiB/s)
Loaded environment from uEnv.txt
Importing environment from SD ...
Copying Linux from SD to RAM...
reading uImage
```

Figure 8.1: Boot of the Linux distribution.

8.4 Time-Tagging Module Testing

To verify that actual time tags are used in the AIT QKD module, the TTM module was tested separately. To verify the timestamps a C++ code was written for get the time tags from the TTM and write it over TCP/IP to a remote PC.

Listing 8.1: Function for preparing time tags to send over TCP/IP to remote PC.

```
1  void Networks::Get_TimeTags() {
2    //....variables init
3    StopTimetags();
4    CloseTimeTag();  //closing...just to be sure
5
6    OpenTimeTag();
7    StartTimetags();
8
9        //clear buffer
10       bzero(tag_buffer1, 780255);
11
12       //run until buffer size is reached
13     while (running_buffer1) {
14       int have = ReadNextTag(&chan, &time);
15
16       if (have) {
17         if (counter == 0) {
18           starttime = time;
19         }
20         //copy timetag and channel information into buffer
21         memcpy(&tag_buffer1[counter * (sizeof(time) + sizeof(chan))], &
       time,
22             sizeof(time));
23         memcpy(
24             &tag_buffer1[counter * (sizeof(time) + sizeof(chan))
25                 + sizeof(time)], &chan, sizeof(chan));
26         counter++;
27
28         if (counter == NUMBEROFCOUNTS) {
29           endtime = time;
30           counter++;
31           char * errortext = GetErrorText(ReadErrorFlags());
32
33           //calculate hz out of event rate
34           double ns = (endtime - starttime) * (78.125) * 1e-9;
35           double hz = (NUMBEROFCOUNTS / ns);
36
37           //copy Hz and error into the end of the buffer
38           memcpy(&tag_buffer1[counter * (sizeof(time) + sizeof(chan))],
39               &hz, sizeof(hz));
40           memcpy(
41               &tag_buffer1[counter * (sizeof(time) + sizeof(chan))
42                   + sizeof(hz)], errortext,
43               strlen(errortext));
```

```
44
45            running_buffer1 = 0;
46          }
47        }
48      }
49    return;
50 }
```

The remote PC received the time tags in a predefined buffer size. The packets received where then compared to results of another TTM (TTM8000 of AIT). Also, single channels where continuously plugged and unplugged to compare the actual action to result the TTM delivered.

8.5 QKD Module Testing

The first tests which were used to verify the integrity of the QKD software were provided with the QKD software by AIT. For the purpose of testing, this testing applications were additionally installed on the SoC and removed afterwards. The tests the AIT provided check the integrity of single modules and the availability of the overall system. To actual use the new developed time-tagging module with the AIT QKD software, the recorded timestamps, obtained by the TTM functions, are forwarded to the QKD-stack module.

The tests which are done on the Xilinx Zedboard:

Listing 8.2: This test output shows the functionality of single modules and the overall pipelining system.

```
 1 ./test-pipeline-sifting
 2 found qkd module running with pid 4379 - killing ...
 3 modules found: 1
 4 stopping modules ...
 5 stopping modules ... done
 6 starting modules ...
 7 started module: bin/modules/qkd-entanglement/qkd-entanglement
 8 started module: bin/modules/qkd-cascade/qkd-cascade
 9 started module: bin/modules/qkd-confirmation/qkd-confirmation
10 started module: bin/modules/qkd-buffer/qkd-buffer
11 started module: bin/modules/qkd-privacy-amplification/qkd-privacy-
       amplification
12 starting modules ... done
13 modules found: 5
14 stopping modules ...
15 stopping modules ... done
16 starting modules ...
17 started module: bin/modules/qkd-entanglement/qkd-entanglement
18 started module: bin/modules/qkd-cascade/qkd-cascade
19 started module: bin/modules/qkd-confirmation/qkd-confirmation
20 started module: bin/modules/qkd-buffer/qkd-buffer
21 started module: bin/modules/qkd-privacy-amplification/qkd-privacy-
       amplification
22 starting modules ... done
```

Listing 8.3: AIT QKD software runs with the modified TTM module.

```
./test-mod-entanglement
found qkd module running with pid 5207 - killing ...
qkd key generation setting:
  file:              cat_keys
  keys:              1000
  first id:          1
  size:              2048
  rate:              0.05
  exact:             0
  zero:              0
  set error bits:    0
  disclosed bit rate: 0
  quantum:           1
created key #1
created key #2
created key #3
.
.
.
created key #999
created key #1000
waiting for module(s) 'cat' to be stalled ...
waiting for module(s) 'entanglement' to be stalled ...
```

8.6 Summary

Once the single modules are set up, Commissioning can be done without much effort. Even after changing the setup, the QKD software can run on the developed QKD Module. Changes in the software can be updated via the Yocto Project and simply replacing the image of the Linux distribution on the SD-Card. The TTM on the FPGA has been tested separately by streaming the time tags over a TCP/IP connection to a remote PC for a manual analysing and verifying the coincidences. The QKD software consists of a pipelining system, in which each software module can be tested separately. The functionality of each module has also be verified by running the test-mod-entanglement module successfully.

Chapter 9

Conclusion and Outlook

QKD is still in the early stage of development and probably still need a few years for commercial implementation. Although QKD is theoretical secure, practical problems prevent QKD from being useful. Various QKD protocols exist with their individual advantages and disadvantages. A promising protocol is the entanglement protocol which uses entanglement as distribution to both communication parties. This gives an big advantage for commercial use, since the actual source for creating and distributing the quantum key can be placed on a satellite for a wide range of users. Until now, no entanglement on a satellite has been implemented and therefore remains to be proven.

This thesis demonstrated a working platform for QKD on a single SoC. The module was divided into TTM and the Linux operating system. The FPGA was configured as an TTM to meet the strict time constraints which are needed for QKD. The basic functionality of a TTM might be simple but have to be efficient. All TMMs share their common purpose and can be used over a wide variety of QKD applications. Yet, TTMs standalone system exist, a SoC with an explicit TTM is not available yet. This might be feasible in the future to save resources on the FPGA.

At the same time, the ARM cores run a customized Linux distribution primary for handling the AIT QKD software and the the network connections. The fully customized Linux distribution was created by using the meta-layer for Xilinx SoC devices to gain access to the wide variety of peripherals, which might be used in the future, of the Zedboard. Furthermore, the created recipe for the AIT QKD gives a big advantage for further compatibility after likely changes of the AIT QKD software. Also, by using the Yocto Project, compatibility of various SoC and hardware platforms have been achieved.

By using the AIT QKD software, the Linux distribution is now able to create

coincidence tables and create an quantum key for quantum key encryption. This includes sifting, error correction confirmation and privacy amplification. After this processing the key is used for a bitwise XOR-operation for any type of information or files.

The QKD software runs on the SoC but no performance test has been made. The actual count and coincidence provided by the source was handled well so that no actual lack of performance was noticed. QKD can be performed but for actual commercial usage, the quantum key generation rate is too low for an comfortable usage. Further implementation have either be more efficient or better hardware is needed for quantum key distribution for large file sizes.

Beside practical problems with optical precision and information loss during the distribution, a new step has been made on the software level.Time-Tagging of quantum events remains a sensitive issue, but with an implementation on an FPGA on a SoC device usability has been increased. A module which is capable of record this events and create a quantum encryption key has been developed. New users can focus on QKD and do not have to have specific knowledge about the TTM and how timestamps can be accessed or forwarded. The recipe for the Yocto Project offers are wide variety of common tools and software to easily adapt to personal preferences and needs. Although, the AIT QKD is still under heavy development, future changes are going to be adapted with no or little effort.

A next step in the usability of QKD is testing and improving of performance. As the communication rates are rising, the keys created by the software also have to raise. Currently, the rates are most likely limited by the number of entangled photons the source can provide, but post processing for embedded devices which are current needs a good amount of resources to keep up with the key generating processes. Also, further error correction process might be added. Another idea would be, to set up a web service which tracks the QKD data 24/7 and shares it for future developing in QKD. Especially specialists in IT security might be interested in such a functionality.

References

Literature

[1] Charles H. Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing* 1 (1984), pp. 175–179 (cit. on p. 1).

[2] Hannes René Böhm. *A Compact Source for Polarization Entangled Photon Pairs*. Tech. rep. Vienna, Austria: Atominstitut der österreichischen Universitäten, 2015 (cit. on pp. 12, 13, 16–18, 22).

[3] D. Deutsch et al. "Quantum privacy amplification and the security of quantum cryptography over noisy channels". Version 1. In: (1996), pp. 1–13. URL: http://arxiv.org/abs/quant-ph/9604039 (visited on 07/15/2015) (cit. on pp. 16, 17).

[4] Albert Einstein, Boris Podolsky, and Nathan Rosen. "Can quantum-mechanical description of physical reality be considered complete?" In: *Physical review* 47.10 (1935), p. 777 (cit. on pp. 3, 10).

[5] Artur K. Ekert et al. "Practical quantum cryptography based on two-photon interferometry". In: *Phys. Rev. Lett.* 69 (9 Aug. 1992), pp. 1293–1295. URL: http://link.aps.org/doi/10.1103/PhysRevLett.69.1293 (cit. on p. 17).

[6] R.P. Feynman, R.B. Leighton, and M.L. Sands. *The Feynman Lectures on Physics*. The Feynman Lectures on Physics Bd. 1. Addison-Wesley, 1963 (cit. on pp. 3–5).

[7] Daniel Genkin, Adi Shamir, and Eran Tromer. "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis". In: (). URL: https://eprint.iacr.org/2013/857.pdf (visited on 09/01/2015) (cit. on p. 1).

[8] Johannes Handsteiner. "A transportable polarization-entangled photon source". Version 1. In: (2014) (cit. on p. 22).

[9] Caleb Ho, Antia Lamas-Linares, and Christian Kurtsiefer. "Clock synchronization by remote detection of correlated photon pairs". Version 1. In: *New Journal of Physics* 11 (2009), pp. 1–14. URL: http://arxiv.org/abs/0901.3203 (visited on 03/27/2015) (cit. on pp. 21, 23).

[10] Texas Instruments. *LVDS Application and Data Handbook*. URL: http://www.ti.com/lit/ug/slld009/slld009.pdf (cit. on p. 22).

[11] Xiao-Song Ma et al. "Quantum teleportation over 143 kilometres using active feed-forward". In: *Nature* 489 (2012), pp. 269–273 (cit. on pp. 19, 27).

[12] Oliver Maurhart. *The AIT QKD Software Handbook* (cit. on p. 39).

[13] Mary Øystein et al. "Error Estimation, Error Correction and Verication In Quantum Key Distribution". Version 1. In: (2012), pp. 1–9. URL: http://arxiv.org/abs/1210.6520 (visited on 07/15/2015) (cit. on p. 16).

[14] Suetonius. *Life of Julius Caesar*. AD 121 (cit. on p. 12).

[15] Jackson Labs Technologies. *LC-XO-Plus Low Cost Desktop GPSDO Module Kit Specification*. URL: http://www.jackson-labs.com/assets/uploads/main/LC_XO-PLUS-specsheet1.pdf (cit. on p. 35).

[16] Austrian Institute of Technology. *TTM8000*. URL: http://www.ait.ac.at/uploads/media/Datasheet_Project_TTM8000_EN_V1_02.pdf (cit. on p. 20).

[17] Till Till Zoppke and Christian Paul. *FAKTORISIERUNG GROßER ZAHLEN MIT EINEM QUANTENCOMPUTER*. 2002. URL: http://page.mi.fu-berlin.de/alt/vorlesungen/sem02/shors_algorithm.pdf (cit. on p. 1).

[18] HALLDEN U. "An explanation of haidinger's brushes". In: *A.M.A. Archives of Ophthalmology* 57.3 (1957), pp. 393–399. URL: http://dx.doi.org/10.1001/archopht.1957.00930050405011 (cit. on p. 6).

[19] Daqing Wang. "Free-space quantum teleportation over 143-Kilometres". In: () (cit. on p. 23).

[20] Daqing Wang. "Timing with correlated photon pairs for space-ground quantum communication applications". In: () (cit. on p. 21).

[21] Henning Weier. "European Quantum Key Distribution Network". Version 1. In: (2011) (cit. on p. 17).

[22] W. K. Wootters and W. H. Zurek. "A single quantum cannot be cloned". In: *Nature Publishing Group* 299 (1982), pp. 802–803 (cit. on p. 9).

[23] Xilinx. *Avnet Product Brief Zedboard*. URL: http://zedboard.org/sites/default/files/product_briefs/ZedBoard%20Brochure%20(English).pdf (cit. on pp. 32, 33).

Online sources

[24] European Space Agency (cit. on p. 28).

[25] National Instruments. *What Clock Error Means to Your Measurement System*. July 2015. URL: http://www.ni.com/product-documentation/4801/en/ (cit. on p. 20).

[26] National Instruments. *What Is Meant by the Stability of an Onboard Clock?* July 2015. URL: http://digital.ni.com/public.nsf/allkb/486E54911E7D1376862566B4007046B4 (cit. on p. 20).

[27] *Lumière polarisée*. Aug. 2015. URL: http://http://fr.academic.ru/dic.nsf/frwiki/1086070 (cit. on p. 7).

[28] Lewin Mäser (cit. on p. 6).

[29] Oliver Maurhart. *Austrian Institute of Technology Quantum Key Distribution R10 Software*. July 2015. URL: https://git-service.ait.ac.at/quantum-cryptography/qkd (cit. on pp. 37, 38, 40).

[30] *Polarisationssplitter*. Aug. 2015. URL: http://www.itwissen.info/definition/lexikon/PBS-polarizing-beam-splitter-Polarisationssplitter.html (cit. on p. 10).

[31] *The Yocto Project*. July 2015. URL: www.yoctoproject.org (cit. on p. 37).