FH | JOANNEUM
University of Applied Sciences

Construction and Evaluation of a Non-Deterministic Random Bit Generator with Mobile Sensors as Entropy Source

Master's Thesis
submitted in conformity with the requirements for the degree of
Master of Science in Engineering (MSc)
Master's degree programme **IT & Mobile Security**

FH JOANNEUM (University of Applied Sciences), Kapfenberg

supervisor: Dr. Wilhelm Zugaj
submitted by: Stefan Kutschera, BSc
personal identifier: 1710419010

May 2019

Formal declaration

I hereby declare that the present master's thesis was composed by myself and that the work contained herein is my own. I also confirm that I have only used the specified resources. All formulations and concepts taken verbatim or in substance from printed or unprinted material or from the Internet have been cited according to the rules of good scientific practice and indicated by footnotes or other exact references to the original source.

The present thesis has not been submitted to another university for the award of an academic degree in this form. This thesis has been submitted in printed and electronic form. I hereby confirm that the content of the digital version is the same as in the printed version.

I understand that the provision of incorrect information may have legal consequences.

Kapfenberg, Austria, May 2019

Stefan Kutschera, BSc

Abstract

The aim of this Master Thesis at hand is to construct and evaluate a Random Number Generator (RNG), based on entropy sources available within smartphones, which does not only produce randomness but is also competitive to other existing and already proved RNGs.

To answer if it is possible to create such an RNG, a prototype with the proposed algorithm, which is also backed up by the state of the art, was implemented. The outcome of the prototype implementation was then tested against statistical vulnerabilities with the Statistical Test Suite from NIST.

On one hand the results of the cosmic ray experiment were strong but had severe failures during execution and unexpected environmental complications while on the other hand the implementation based on gathered audio and video sensors were successful with some exceptions. These exceptions include the unsolved non-randomness for the video as entropy source on some other test devices as well as a low random extraction within the audio entropy source.

However, the results showed that the proposed algorithm on entropy sources within a smartphone are competitive in most cases but need more improvement and a more extended investigation on the non-random output on some devices. To achieve a true non-deterministic RNG the implementation of the ERBG should additionally use existing smartphone based muon detection. In order to fulfill the research question regarding the creation of a non-deterministic RNG the combination of entropy sources such as audio and video as well as muon detection should be considered.

Acknowledgement

First of all I would like to thank my supervisor Dr. Wilhelm Zugaj at FH JOANNEUM Kapfenberg as he had not only allowed this master thesis at hand to be my own work, he guided me always in the right direction when I was in need of guidance. Moreover, I really enjoyed and appreciated the discussions about the field of this thesis.

I also would like to thank Mag[a]. Petra Maria Kletzenbauer at FH JOANNEUM Kapfenberg for not only proof-reading this Master Thesis but also being open for semi private discussions on personal matters and future dreams like living and working in San Diego, California.

The initial mentoring of DI (FH) Günther Hutter, MSc in my pre academic stage has unleashed my power to finally start into my academic career, I'm very thankful for that.

As I had to gain and improve my English skills I replaced broadcast radio with the podcast series "BBC Inside Science" with Dr. Adam Rutherford. Whenever there was a chance I turned on the podcasts and learned a lot, not only by listening to the high diversity of scientific topics but also to the English spoken within each podcast. Without Dr. Adam Rutherford and his postcast series I would not have had such a strong learning curve, thank you.

I also would like to thank my parents not only for keeping faith in me but also for keeping their vow for each other and being responsible for the family.

Finally, I must express my biggest and profound gratitude to my best friend and wife, Marie-Theres, as it was not only possible to consistently strengthen the bond between us but also pursue common life goals besides my academic career. As we vowed, for better, for worse, for richer for poorer, in sickness and health even beyond death.

Thank you to all of you who have mentored me, being patient with me, loved me unconditionally and encouraged me to become the men I have become and continue to become.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

> *"Die Quantenmechanik ist sehr achtunggebietend. Aber eine innere Stimme sagt mir, daß das noch nicht der wahre Jakob ist. Die Theorie liefert viel, aber dem Geheimnis des Alten bringt sie uns kaum näher. Jedenfalls bin ich überzeugt, daß der nicht würfelt."*
>
> [Albert Einstein (Einstein, 1926)]

One of the main focuses of cryptography is about secure communication. Encryption is when a plaintext and a key are forged into a chiphertext which can be decrypted by a symmetric- or asymmetric key. Information should be unreadable to anyone without a proper key. Excellent cryptography is when the algorithm and methodology of the encryption process is available without any restrictions because everyone can proof or disproof its strength. With an publicly available encryption algorithm a great weakness lies within the used random input.

Albert Einstein acknowledged in a letter to Max Born, a German Mathematician and Physicist, that 'he', thus God, does not play dice. In fact, Einstein predicted theories about how the universe should work, which were later proven to be correct. In this particular matter it seems, based on what we know today, that he was not right at all. For instance, nowadays quantum mechanics is used to create a chain of numbers which has such a long sequence of no repetition that the output, a chain of random numbers, is not vulnerable against statistical tests.

1.1 Problem Statement

The concept of a so-called Quantum Random Number Generator QRNG is relatively easy available not only to businesses but also to private individuals. With a price tag of 1.200 USD and being quite huge in size a QRNG does not fit in our pockets and not everyone can afford the time and money to possess a strong and trustful source of a Random Number Generator (RNG). This fact was the inspiration of my Master Thesis to develop a RNG. A RNG based on everyday carry-on items like our smartphones. Already available sensors on our smartphones could be used to receive the necessary entropy source, for example external input to generate a random output. Such highly available random number generators could function as a base for secure communication. Hence, a secure ciphertext does not only need a strong algorithm, it also needs a strong source of randomness.

1.2 Research Questions

The master thesis should be able to answer the following questions:

- How can a Non Deterministic Random Bit Generator be constructed by using smartphone sensors as entropy source based on existing approaches?
- How does the evaluation of the proposed algorithm compare to existing Random Bit Generators?

1.3 Hypothesis

Random number generators play a huge role in our everyday life, even though we do not recognize them. Behind a secure connection to a website, for example, there is often an RSA encryption which uses a random number for generating a key. Hence, it is important to have strong and cheap random number generators at hand. In order to address this problem, this Master Thesis will try to create a prototype of a widely available RNG that can be used to create a strong random sequence which is not vulnerable against statistical attacks. A successful result of this Master Thesis could also make the electronic exchange of messages more secure by providing a prototype of a cheap but strong, thus economical, source of a RNG.

As a proof of concept a simple application for smartphones is created that implements the Economy Random Bit Generator (ERBG). This application is able to use different or a single entropy sources, like for instance the camera and microphone of a smartphone. By starting the gathering process a video and/or audio is captured and stored internally. Another process , with the proposed algorithm in this Master Thesis, extract the randomness of the so-called raw file into a newly created file. This new file containing the extracted randomness consist of only zeros and ones. The extracted random file can be used in another work by building a framework around the ERBG, so that the encryption process of the following would be possible.

Further implementation of the ERBG would interpret random bits to random numbers, for instance 'A' is represented as '065' in the ASCII-Code table, which is '01000001' in binary. Therefore, the output string '01000001' from the random bit generator would finally represent 'A' in the Random Number Generator. As a result, neither the output of an Random Bit Generator is vulnerable against statistical attacks nor the key gathered from such source. By using this strong key it is possible to encrypt plaintext to a ciphertext which can be sent as a message through highly insecure applications and networks. Furthermore, both clients need a common set of keys which need to be exchanged once.

A message sent through unsecured infrastructure could look like this:

Name	**Value**
First Message	'Use key number 3 on sheet 1'
Second Message	'123AB456CD321DC'

Table 1.1: Pseudo messages that Alice will get from Bob

Table 1.2, is a proposal of the prerequirements of what is needed for Bob[1] to send Alice a secret message.

Bob knows that his network is permanently monitored and sends two messages to Alice, as can be seen in Table 2.2. First message in plaintext, the second in ciphertext which only Bob and Alice should be able to read.

As only Bob and Alice have those secure keys in their property, Alice can now use key number 3 from sheet 1 which is '111222' to decipher the message from Bob. As a possible digital intruder like Eve and Mallory, they have to use brute force as it is

[1] Alice, Bob, Eve and Mallory are fictive characters commonly used in Cryptography to explain certain actions

Name	Value
Plaintext	'Will you meet me today at our coffee bar at 2 pm? Yours Bob'
ERBG Number	'01011001011101011101110011010101111000001101 01'
Generated Keys	'Sheet 1: '123456; 654321; 111222; 333444; 555666"
	'Sheet 2: '123456; 987654; 999888; 777666; 456789"
Ciphertext:Key pair	'111222':'123AB456CD321DC'

Table 1.2: Pseudo prerequirements for a message Bob will send to Alice

impossible to recalculate the key because they used a strong RNG source, which is ERNG.

1.4 Methodology

As of today, there are many other RNGs available, some of them do also use sensors of smartphones as their source of entropy. The most relevant one for this Master Thesis are included. Those existing RNGs are discussed in Chapter 2.7. One of them examines the quality of randomness throughout every available sensor within smartphones by also using different methods, the others specialised on video and/or audio as an entropy source. Most of them use a residue class calculation, also called modulo operation, where two of them use $b \equiv \pmod 2$. In addition, most of them operate on a single color channel.

The Master Thesis uses the knowledge and experience based on existing literature and provides an approach that is not only limited to single picture frames but uses the entropy within a video file. Moreover, the method and proposed algorithm can be extended and used outside the setup of video and/or audio as an entropy source.

Chapter 2

Related Work

This chapter addresses the state of the art regarding random number generators, the historical background and informs about common domain-specific terms and standards. Finally, it describes what random stands for in detail.

2.1 Randomness in History

According to Batanero, Green, and Serrano, 1998, the term random itself appeared in the Middle Ages and was attributed to supernatural forces. It was the same time when random devices such as dices or astragali (i.e. bones used as an oracle) were used to predict the future and prevent any advantages of participants in games: “Nothing happens at random; everything happens out of a reason and by necessity”, a quote from the Greek philosopher Leucippus in the 5th Century B.C. (Bennet, 1993). Poincare, 1936, on the other hand, distinguished random phenomena about the probability that a calculus can give some information and random phenomena for which there was no possibility of prediction until the law behind was fully discovered.

2.2 Definition of Randomness

As Batanero, Green, and Serrano, 1998, argue, the term random can be divided into two groups, formal and informal.

From an informal point of view ’random’ is used to explain too complex causes or causes that are not fully explainable by known physical laws. From the formal point

of view, the term 'random' is the lack of predictability or patterns. For an example of predictability every tenth result of a coin tossing experiment is collected. If the list of coin tossing results is analyzed, it shows that the probability for the head is $\frac{2}{3}$. Someone can take advantage of this fact and bet on the head. Someone could take advantage of this fact by simply putting the bet on the head which leads to that person winning $\frac{2}{3}$ of the time.

On the other hand, the formal definition of random is when a sequence has a maximum complexity or in other words, the lack of patterns within the sequence, thus the minimal number of signs necessary to codify a sequence. For example, '0101010101010101' can be compromised as 8 times '01' (Batanero, Green, and Serrano, 1998).

Hence, it can be argued that firstly random is a sequence which can not be predicted. Secondly random has a maximum of information in included, this concludes that it cannot be compressed.

2.3 Examples of Non-Randomness

At first glance coin tossing may seem random but with a closer look it can be argued that with certain circumstances given the outcome of a single coin toss can be predicted with a very high amount of certainty. As Holmes, Diaconis, and Montgomery, 2004, point out in their paper that if the angle ψ between $\overrightarrow{M}$ and the normal to the coin is less than 45 degree the coin never turns over. This can be visualized in Figure 2.1 (Holmes, Diaconis, and Montgomery, 2004).

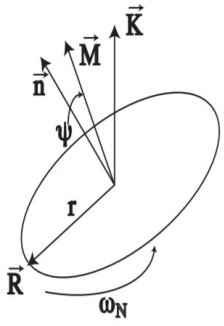

Figure 2.1: Coordinates of Processing Coin (Holmes, Diaconis, and Montgomery, 2004)

As a result, an apparatus was built which took the angle and other parameters in place. The outcome showed that the coin, when started heads up, always landed heads up. Holmes, Diaconis, and Montgomery, 2004, were not the first who made this discovery.

Similar experiments where done by Poincare, 1936, with the analysis of roulette and were later continued by Eberhard Hopf who analyzed Buffon's needles. The Buffon's needles are about the approximation of π. In the progress, a set of similar sticks is dropped randomly on a surface with parallel lines, then the used sticks were multiplied with two and divided by the sum of each stick which crossed a line (Hopf, 1934), (Hopf, 1936), (Hopf, 1937), (Holmes, Diaconis, and Montgomery, 2004).

2.4 Physics Behind this Work

As we learned what random meant in the Middle Ages and how the term 'random' is defined today, we have also seen that random, from an informal point of view, does not know the laws behind a particular event. However, there are some sources for randomness from events that are fully understood. The idea behind it is that we take advantage of external sources. Taking the perspective of software development external sources are considered in order to not use java.random or any other built-in Pseudo Random Number Generator (PRNG). The problem with those Pseudo Random Number Generators is that when they are fed with the same seed[1] their output will be the same random numbers as before. Therefore, built-in Random Number Generators are deterministic, the same output can be expected every time if an apparatus or algorithm is fed with the same input. Hence, this work should cover a vast majority of physics used in respect of sound, visualization and quantum mechanics as it is necessary to understand the interconnectedness of certain physic events and behaviors.

2.4.1 Sound and Audio

The three components of the human ear, that made 'hearing' possible, are the outer, middle and the inner ear. The sound of our surroundings are tunneled through the ear canal to the middle ear in which the tympanic membrane, or the so-called eardrum, sits. The sound waves colliding with the eardrum are amplified by three tiny bones that act as a lever. The vibrations of that process then travel to the cochlea. The cochlea is a fluid-filled coil that holds small hair, those hair are attached to the nervous system. Here the vibration finally is converted into electrical signals which travel to the brain and is processed by it, see Figure 2.2. Roberts-Breslin, 2012, also mentions that the sound in our brain is processed in three different categories, such as speech, music and background noise which is stored in the short-term memory.

[1] A seed in this context is the initial value by which the RNG is started

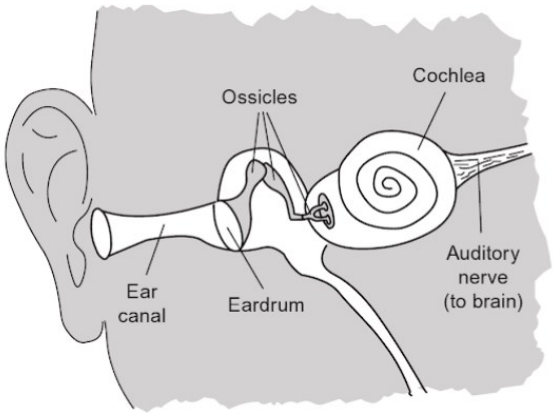

Figure 2.2: The sound waves have to travel through various stages to reach the brain (Roberts-Breslin, 2012)

This biological concept of the human ear is almost the same concept within microphones. Microphones also convert vibrations from sound waves into electrical signals but there are two different approaches: one approach is to work with magnetism, where a small coil is placed near a magnet which generates a magnetic field. When sound waves reach the coil the coils start to move in correlation with the sound waves amplitude and frequency. This movement of the coil creates electric energy, voltage, which retains the information of the original soundwave. That approach is used in so-called

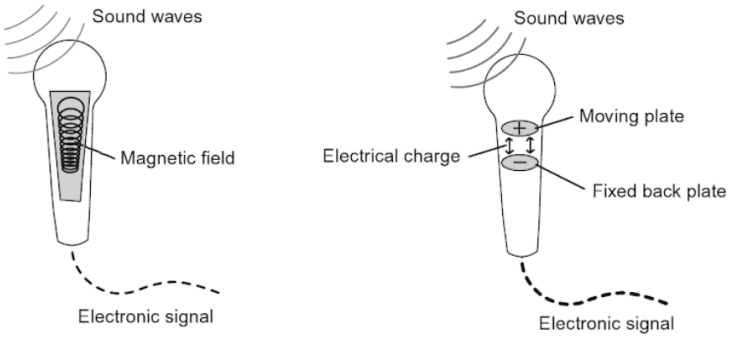

Figure 2.3: Schematic portrayal of a dynamical based (left) and a condenser based micro (Roberts-Breslin, 2012)

dynamical microphones.

On the other hand, there is a condenser microphone. The condenser or capacitor microphone has two plates, one fixed and one moving, one with a positive charge and the other with a negative charge. The sound waves are cause the not fixed plate to move. Due to that relation and the change in proximity of the two plates the voltage constantly changes. The closer the two plates the higher the voltage (Roberts-Breslin,

2012).

With both, the condenser and the dynamical microphone the voltage is so low that it needs to be amplified before being processed (Robjohns, 2010) (Roberts-Breslin, 2012) (Jaros, A. Nussbaumer, and P. Nussbaumer, 2012).

2.4.2 Visual and Video

Now it is clear what randomness meant in earlier stages of history compared to today's definition. The mechanics and laws behind observing our surroundings with the help of hearing and seeing were also described. Moreover, mankind has brought our biological eyes and ears into the digital world. As hearing is about receiving the right resonance of an audio source within a reachable distance, light works in the same way.

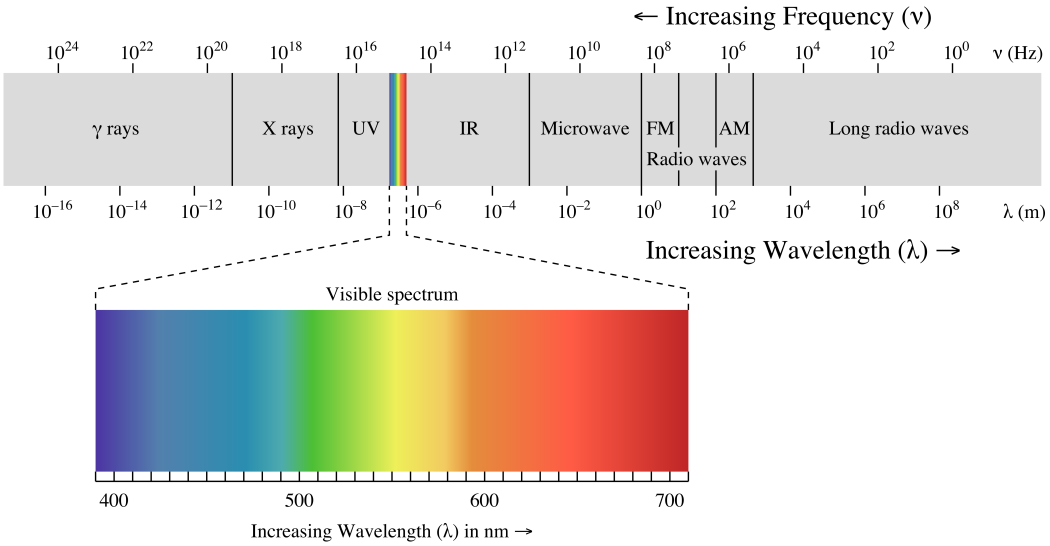

Figure 2.4: Visualization of electromagnetic wavelengths and their categorization (Ronan, 2019)

As described in Chapter 2.4.1, sound is created by a vibrating membrane. It may also be of interest to know how light, i.e. electromagnetic waves as shown in Figure 2.4, is created. Therefore, this Master Thesis also discusses atoms on the Rutherford-Bohr Model as shown in Figure 2.6. An atom has protons, neutrons and electrons. Protons and neutrons are the massive part in the center of atoms, the so-called nucleus. To describe how visible light, thus electromagnetic waves with wavelengths of 400-700 nanometers, is produced, a simply hydrogen atom is taken to describe that process. The hydrogen atom has only one electron within its shell which revolts in one of a limited number of circular orbits around the nucleus of the hydrogen atom. The electron is in

its so-called ground state when it is the closest to the nucleus. On the other hand, there is the state of excitation, in which the electron 'jumps' to an orbit further away of the nucleus of the hydrogen atom. To achieve this state of excitation the hydrogen atom has to absorb a certain amount of energy. For example, this can be done by heating the hydrogen atom. However, as visualized in Figure 2.6, the state of excitation is unstable and the electron releases its energy at a random point and falls back to the ground state. By falling from an excited state to the ground state the electron releases energy in form of a photon. The frequency of the released photon, thus the electromagnetic wave, is in relation to the energy transition. The result of this physical process can be observed as light (H. Smith, 1999), (Bewick et al., 2016).

Figure 2.5: These are the emission spectrum lines of the hydrogen atom in the visible spectrum for human eyes. Note that the wavelengths of the two lines on the very left are less then 400 nanometers and are therefore considered to be ultraviolet but still visible to the naked eye (Homann, 2009).

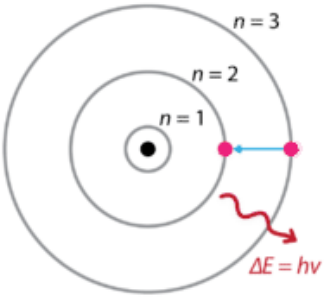

Figure 2.6: Showing the Rutherford-Bohr Model of an atom where an electron releases energy and emits photons by falling from the unstable state of excitation back to the ground state (Bewick et al., 2016).

To see objects they first have to emit electromagnetic waves which, in turn, they receive from a light source. This light source can be terrestrial, in form of an electric light bulb, the fire of burning wood or extraterrestrial like our sun. A source of light usually has a specific spectrum type. Firstly, the continuous spectrum is a form of genuine light from a source with the whole spectrum of wavelengths, as can be seen in Figure 2.7. Secondly, an absorption spectrum is when light travels through a thermally cooler cloud of gas and photons with an appropriate level of energy are absorbed. Although they may be re-emitted, those wavelengths are missing for the observer on the other side of the gas cloud, therefore the absorption spectrum is also called dark line spectrum. Thirdly, an emission spectrum is when a gas is triggered and atoms are brought to a state of

so-called excitation. In that state of excitation those atoms, specific for that gas, start to emit light in a related wavelength. Only those wavelengths are visible in the spectrum of the observer which is why it is also called bright line spectrum. All three of the discussed spectra's can be seen in Figure 2.7, (Fraknoi, Morrison, and Wolff, 2019, p.172-176), (H. Smith, 1999).

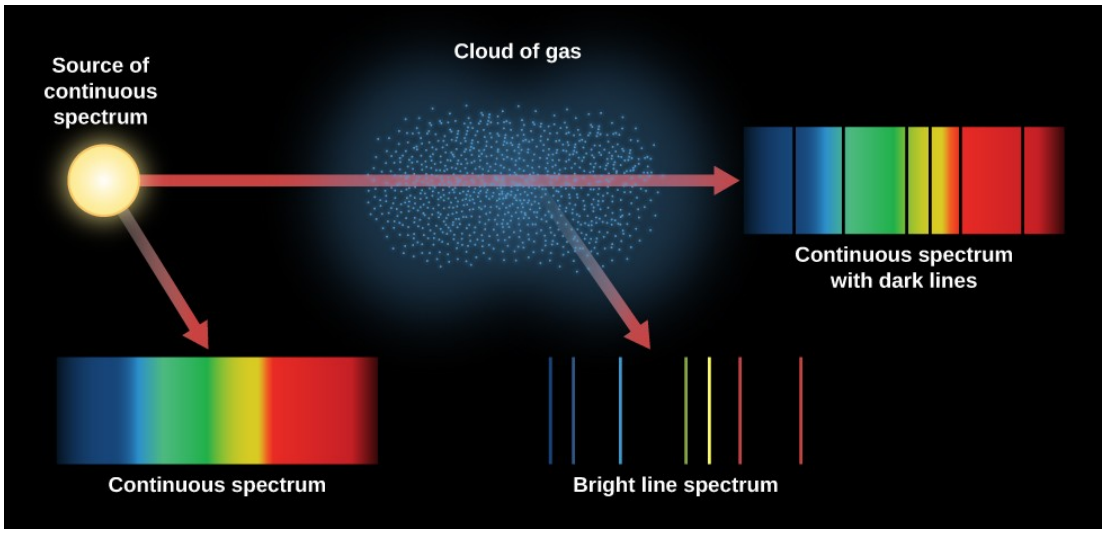

Figure 2.7: This figure shows three principal types of light spectra (Fraknoi, Morrison, and Wolff, 2019, p.175)

To finally see, electromagnetic waves, thus the photons emitted by electrons switching back from the state of excitation to the ground state, have to travel through the cornea and the lens of our eyes, as can be seen in Figure 2.9. When light hits the region called retina within the human eye, the graded electrical activity of photoreceptors are converted to actions potentials which travel through the optic nerve into the brain. The retina is capable of doing so by five different types of neurons, such as photoreceptors, bipolar cells, gallion cells, horizontal cells and amacrine cells as well as neurotransmitters as can be seen in Figure 2.8, (Purves et al., 2004).

When we digitalize the more or less visible, the visual, to a video or photo, those photons emitted by atoms have to be detected. As of today there are two major technologies. On the one hand, the semiconductor Charge Coupled Devices (CCD) and on the other hand, the Complementary Metal Oxide Semiconductor (CMOS). However, in the vast majority and most consumer electronics the CMOS image sensor is used.

The CCD semiconductor image sensor was invented by Boyle and G. E. Smith, 1974, and had, compared with the CMOS sensor, some advantages. Those advantages were a better light sensitivity and less noise. Nowadays the CMOS image senor has improved and surpassed the CCD semiconductor image sensor. In Figure 2.12, a simplified ver-

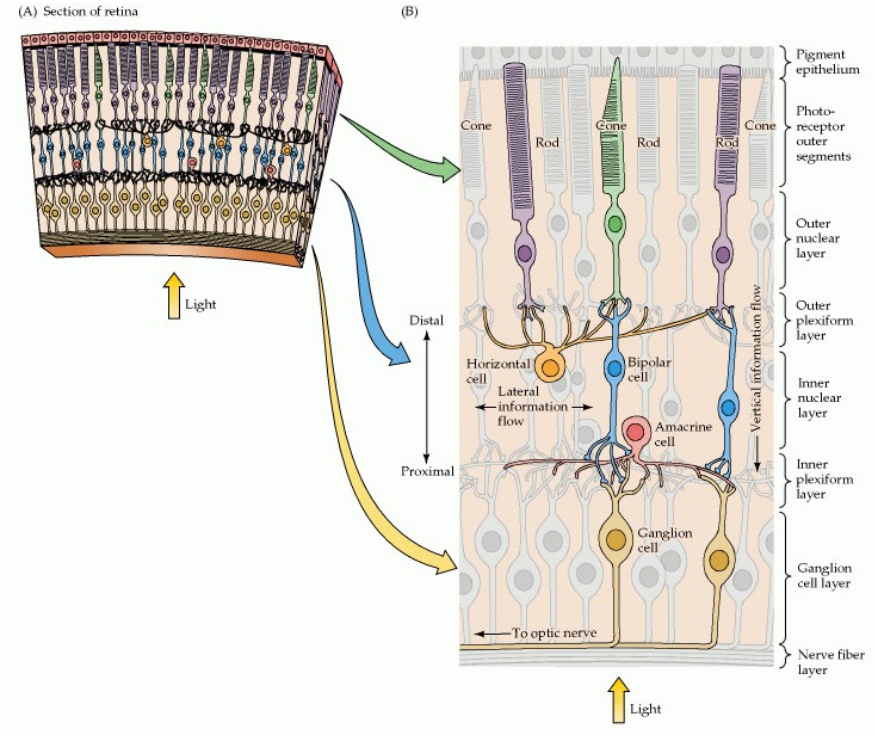

Figure 2.8: Cross section of the retina, showing how light is processed through the layers of five neurons and neurotransmitter (Purves et al., 2004, p.235)

sion of the functionality and the architecture can be seen. Moreover, the disadvantages of CCD sensors are that the analog components require more electronic circuity than the CMOS sensor. Due to the increased power consumption heat issues can have a negative impact on the image quality (AXIS Communications AB, 2010[p.4]).

The CMOS image sensor was invented by Wanlass, 1967, but in its early stages the quality was poor due to its lack in light sensitivity. According to AXIS Communications AB, 2010, and Turchetta, Spring, and Davidson, 2019, this changed rapidly in the recent years.

As of today the advantages of a CMOS sensor are a faster readout, lower power consumption, higher noise immunity and a smaller system size which also lead to better integration possibilities.

However, both share one in common, the color filtering. In order to detect the correct color a filter is laying ontop of the image sensor (see Figure 2.13). Underneath the filter are the photodiodes located. This allows that only specific light, red, green or blue passes through the filter and reaches the photodiodes (see Figure 2.14). This concept is very similar to the light processing of the human eye, where the retina (see Figure 2.8 and Figure 2.10) collects the light and underlying neurons and neurotransmitters response when triggered by a specific wavelength of light. Within the digital world a

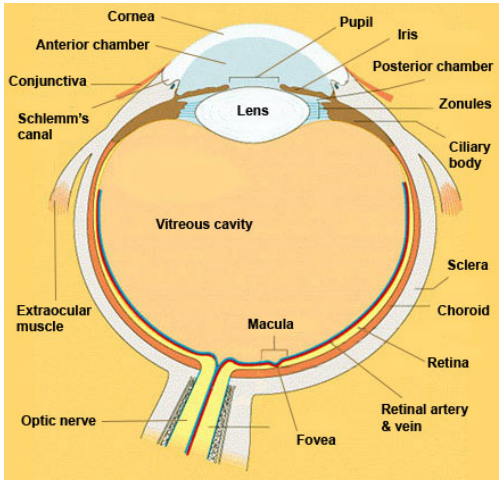

Figure 2.9: Cross section of the human eye (Trobe, 2014)

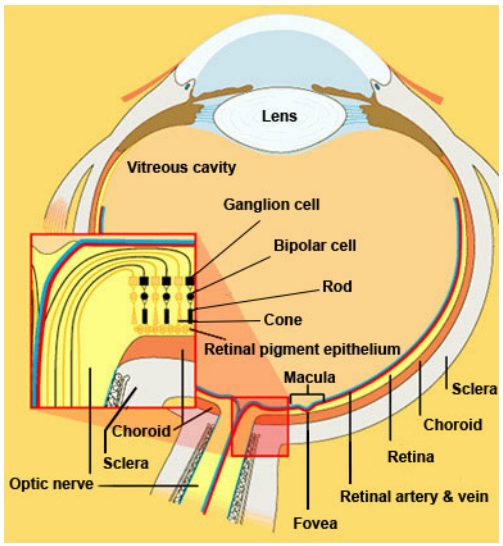

Figure 2.10: Cross section of the retina of a human eye (Trobe, 2014)

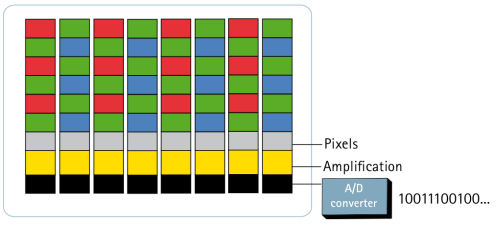

Figure 2.11: This is a simplified overview of the function and architecture of a Complementary Metal Oxide Semiconductor image sensor (AXIS Communications AB, 2010)

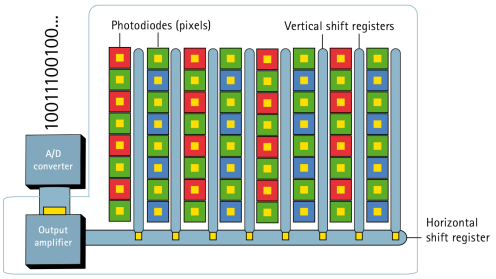

Figure 2.12: This is a simplified overview of the function and architecture of a Charge Coupled Devices semiconductor image sensor AXIS Communications AB, 2010

pattern was invented by Bayer, 1976, this pattern is widely known as the “Bayer Mosaic Pattern”. In order to get a combined picture a process called demosaicing must be done. During the process of demosaicing, the four neighbour-pixels (i.e. horizontal and vertical wise) of each pixel are combined into one pixel with the combined information of red, green and blue (Chang and Tan, 2006). In addition to that, there are several different ways to illuminate the photodiodes within an image sensor. First there is the “Front Illuminated Structure”. This technology is also called frontside illumination and is directly comparable to the function of the human eye but it also has a downside. As light may gets distorted when it travels trough the metal wiring layer it causes lesser light to hit the photodiodes of the image sensor. The other, more newer technology, illuminates the photodiodes directly and sends the proper signal to the wire layer afterwards. The more direct way to illuminate the photodiodes is called “Back Illuminated Structure” or “**B**ack**S**ide **I**llumination” (BSI) (Jeroen Hoerling, 2014).

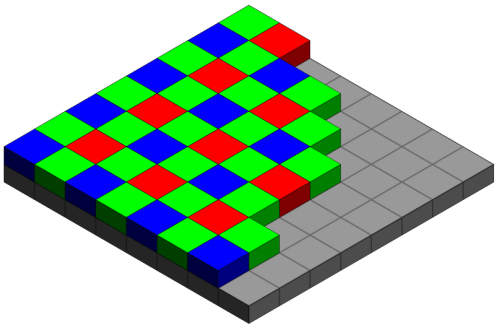

Figure 2.13: In this figure a schematic implementation of a so-called Bayer Mosaic Pattern can be seen with the underlying photodiodes (Burnett, 2006b)

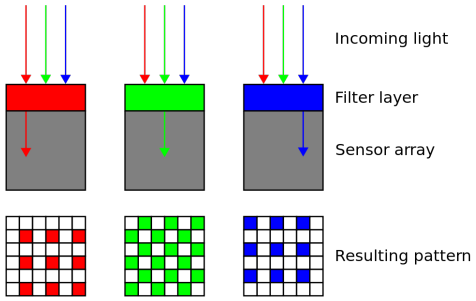

Figure 2.14: A schematic cross section of photodiodes shows how the base colors are processed after they passed the Bayer Mosaic Pattern (Burnett, 2006a)

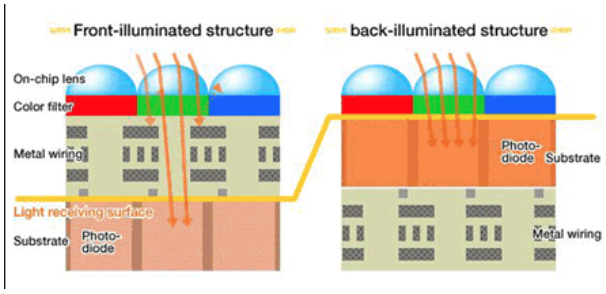

Figure 2.15: Shows two currently used different technologies for the gathering of light within a CMOS sensor (Jeroen Hoerling, 2014)

2.5 Randomization using Cosmic Rays and/or Particles

2.5.1 Quantum mechanics

Quantum mechanics has specific probabilities that come in handy when it comes to setup a secure connection or share a quantum based key. Some of the features of quantum mechanics are explained by the means of photons like the polarisation, unambiguousness of measurement, Einstein-Podolsky-Rosen paradox and the state of entanglement. Photons and electromagnetic waves are both separately describing light.

By polarisation the direction of a electromagnetic field (i.e. electromagnetic waves) is meant. There are different types of polarisation, for instance, linear polarized means that the electric field vector points along a constant direction. Where as at a circular polarization the electrical field vector rotates as the wave propagates. Unpolarized simply means that the electromagnetic wave can come at any polarization. By applying a polarization filter with a certain rotation (e.g. 45 degree or 90 degree), in front of a single photon detector, only those photons are detected. But by applying such filter also a measurement was done.

Before a measurement was taken, the photons are in a multistate. Therefore, a photon is both 45 and 90 degree polarized. The double slit experiment is a very concrete demonstration of that behaviour. As in this experiment a single photon is shot to a wall with two slits, a pattern is created which only exists when two waves interfere with each other. The unambiguousness of measurement means that if a photon is measured its state is fixed, where else before the measurement the photon has hold both states (Bierhorst et al., 2018), (Jennewein et al., 2000), (Suda, 2015).

2.5.2 Muon

There are many particles from space, for instance muons, that we can detect on earth. When high energetic cosmic rays collide with atoms of the upper atmosphere a cascade of 'secondary' particles is produced. Those secondary cosmic rays include electrons and pions, which quickly decay to muons, neutrinos and gamma rays.

Muons reach the earth's surface with an average intensity of about 100 per m^2 with an average lifetime of 2.2 microseconds and a speed of 99.99% of the speed of light, $c = 2,997792458 * 10^8 m/s$, before they spontaneously decay to positrons. Even though thousands of cosmic rays pass through our bodies every minute, the radiation levels

are very low. Nevertheless, the radiation at sea level is only a small percentage of the natural background radiation but rises, the higher the altitude gets. In outer space, without our live-saving magnetic atmosphere, the intensity of radiation is not only a potential hazard to humans in space but also to electronic devices. Impacts of heavily ionizing cosmic ray nuclei can cause small microcircuits to fail or computer memory bits to flip (Mewaldt, Richard A., 2019) (Mehl and Moldover, 1986).

Moreover, Whiteson et al., 2015, developed a smartphone application that is capable of detecting muons on the most sensitive part of a smartphone, the CMOS image sensor. When a muon hits the silicate of the CMOS sensor it can cause a bitflip. A bitflip can easily be detected by completely covering the camera. In order to work correctly the smartphone based muon detector has to be calibrated. The resulting smartphone application called "CRAYFIS" of Whiteson et al., 2015, was tested and confirmed as a smartphone was hit with a muon beam at CERN.

2.6 Random Number Generators

A Random Bit Generator (RBG), often called Random Number Generator (RNG) is required for generating a key or initial vectors in cryptographic algorithms. Random Bit Generators produce sequences of bits like for example '01100'. A Random Number Generator may take this sequence and translate it into a number, in case of '01100' the result would be '12'. However, the term 'Random Number Generator' is commonly used to refer to both concepts.

Random Bit Generators do exist in two different categories: deterministic and non-deterministic. Deterministic Random Bit Generators (DRBGs) use cryptographic algorithms to generate pseudo random bits from an initial value, called a seed. Those deterministic Random Bit Generators are also called Deterministic Random Number Generators (DRNGs) or Pseudo Random Number Generators (PRNGs).

On the other hand there are Non-Deterministic Random Bit Generators (NRBGs), also called True Random Number Generators (TRNGs). The base for a Non-Deterministic Random Bit Generator is an unpredictable physical source that is not controllable by any human. Such an unpredictable source is commonly known as an entropy source (Barker, 2016), (Barker and Kelsey, 2015), (Turan et al., 2018), (Barker and Kelsey, 2016), (Rukhin et al., 2010).

2.6.1 Pseudo Random Number Generator

A PRNG, as mentioned in Section 2.6, is deterministic, which means when given the same input a PRNG always gives the same output. For example for each round PRNG *java.util.Random* gives exactly the same output as the same seed was given (see Listing 2.1). The Listing 2.2 shows that the object *java.util.Random* is initialized four times in separate objects but fed with the same seed. Even though the fed objects are different for each run, they hold the same integer value and each round the four *java.util.Random* instances produce the same random number. The method, "generatePseudoRandomNumbers" as shown in Listing 2.2, is called twice to receive more clarification on the fact that even a second run generates the same random numbers as on the run before.

```
1  Round 0: 1954810208    1954810208    1954810208    1954810208
2  Round 1: -994930201    -994930201    -994930201    -994930201
3  Round 2: 83929109      83929109      83929109      83929109
4  Round 3: -262254147    -262254147    -262254147    -262254147
5  Round 4: 528049954     528049954     528049954     528049954
6  ------------------------
7  Round 0: 1954810208    1954810208    1954810208    1954810208
8  Round 1: -994930201    -994930201    -994930201    -994930201
9  Round 2: 83929109      83929109      83929109      83929109
10 Round 3: -262254147    -262254147    -262254147    -262254147
11 Round 4: 528049954     528049954     528049954     528049954
12 ------------------------
```

Listing 2.1: Output of using java.util.Random PRNG feeded same seed

```
1  import java.util.Random;
2
3  public class Main {
4
5  public static void main(String[] args) {
6  generatePseudoRandomNumbers(120817);
7  generatePseudoRandomNumbers(120817);
8  }
9
10 private static void generatePseudoRandomNumbers(int seed) {
11 Random rnd1 = new Random(seed);
12 Random rnd2 = new Random(seed);
13 Random rnd3 = new Random(seed);
14 Random rnd4 = new Random(seed);
15
16 String delimeter = "\t";
```

```
17
18      for (int i = 0; i < 5; i++) {
19      System.out.println("Round "+i+": "
20      +rnd1.nextInt()+delimeter
21      +rnd2.nextInt()+delimeter
22      +rnd3.nextInt()+delimeter
23      +rnd4.nextInt());
24      }
25      System.out.println("------------------------");
26      }
27      }
```

Listing 2.2: Example of PRNG using java.util.Random feeding same seed

2.6.2 True Random Number Generators

In relation to Section 2.6 a Non-Deterministic Random Bit Generator or True Random Number Generator needs an external source that is not controllable by humans. One example for a True Random Number Generator is provided by ID Quantique. As mentioned in Section 2.5.1, the so-called Quantum Random Number Generator from the company ID Quantique shoots photons, thus light particles, through a semi-transparent mirror and detects them afterwards. Within this setup the events caused by a single photon are either a reflection or a transmission. Both events occur with a probability of 50%. A sketch of such setup can be seen within Figure 2.16 (ID Quantique, Swiss Quantum, 2016).

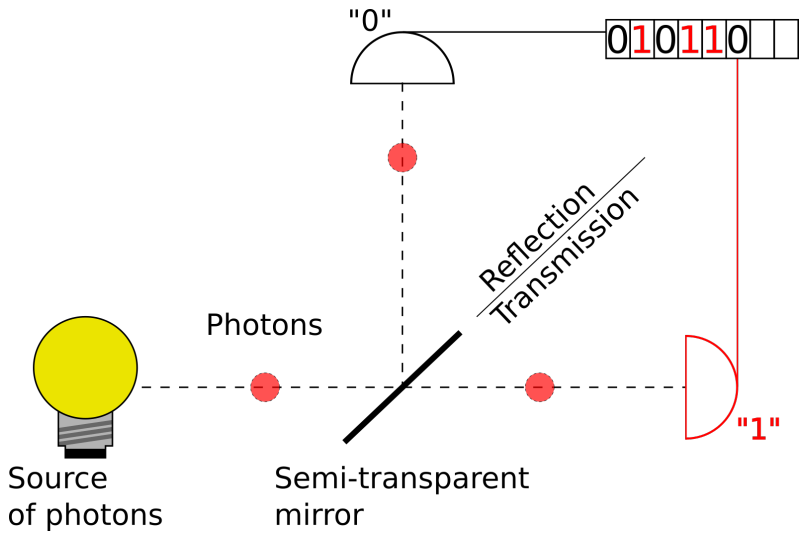

Figure 2.16: Schematic functional setup of the semi-transparent mirror within the QRNG

Figure 2.17: Output from Quantis a QRNG from ID Quantique with the QRNG itself on the keyboard

```
1 10101001110110111010011101010001000010111001001111100111111101011
2 00001110111010011010101010001010000011110000011001011110110000001
3 10111011011010010111010111101011010011110010010111011101001011110
4 01101000100011111000000001100101111101111110011111011011110100100
5 11001101111100000100101011110111011111000000101011100000001000101O
6 11001111100101001100111110101010010111100000001110100101101011111
```

Listing 2.3: Output of using the QRNG Quantis from ID Quantique

2.7 Existing relevant Random Number Generators

This section describes the most relevant existing RNGs in respect to the selected entropy source from audio and/or video sources. Most of them are tested with the NIST Statistical Test Suite from Rukhin et al., 2010. The vast majority of them calculates with residual classes, thus the modulo operation. Like Zhang et al., 2012, Krhovják, Švenda, and Matyáš, 2007, use $\pmod 2$, thus $0 \equiv b \pmod 2$ and $1 \equiv b \pmod 2$ for determining zeros and ones bits used as resulting random bit stream or file. All of them make use of the different base color channels red, green and blue. On one hand Zhang et al., 2012, uses only the red channel as grey scale. On the other hand BlueRand from Leschiutta, 2015, uses only the blue channel.

2.7.1 Proposed RNG by Zhang et al., 2012

Zhang et al., 2012, have proposed and designed a TRNG by using the smartphone camera. In the Figure 2.18, the experimental setup of his proposed TRNG can be seen. For his test a commercial smartphone from Samsung model i9100 with the SONY IMX105 image sensor was used. The configuration of the image sensor can be found in Table 2.1. In order to generate random numbers a finger had to be placed on the camera lens. The placed finger not only had to cover the camera lens but also had to cover the flash light. This was necessary to attenuated the flash light by the skin and muscle tissues.
Zhang et al., 2012, also states that the Bayer Mosaic Pattern with the postprocessing demosaicing process is harmful to randomness. Therefore, they only used the red color pixels for a grey scale. The equation for his final output of 'entropy enhanced true random bits' T can be found in the Equation 2.1. The resulting bitmap created out of zeros and ones from his proposed TRNG can be seen in Figure 2.19(right), in direct comparison to the input image (Figure 2.19(left)).

$$T_j = \pmod 2 \left(\sum_{i=1}^{m} R_i E_{ij} \right) \tag{2.1}$$

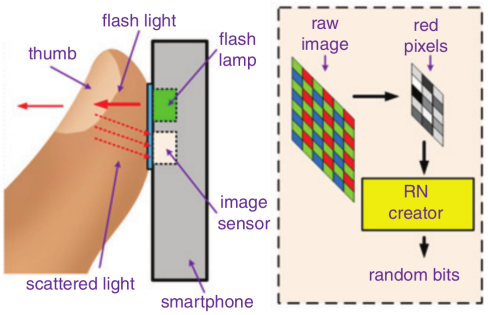

Figure 2.18: Experimental setup of the execution and schematic function of the proposed TRNG from Zhang et al., 2012

Parameter	Setting
Focus mode	Macro
Resolution (pixels)	3264 x 2448
Output bits per pixel	8
ISO	800
Shooting rate (frames/s)	3
Exposure time (ms)	33.3
White balance	Off
Auto contrast	Off
Image format	Bitmap

Table 2.1: Used configuration of the image sensor by Zhang et al., 2012 for his proposed TRNG

2.7.2 Proposed RNG by Leschiutta, 2015

In the work of Leschiutta, 2015, a complete system for secure communication is described. His intent was to send encrypted messages through different message services like WhatsApp, FacebookMessenger, EMail or SMS. The proposed system required

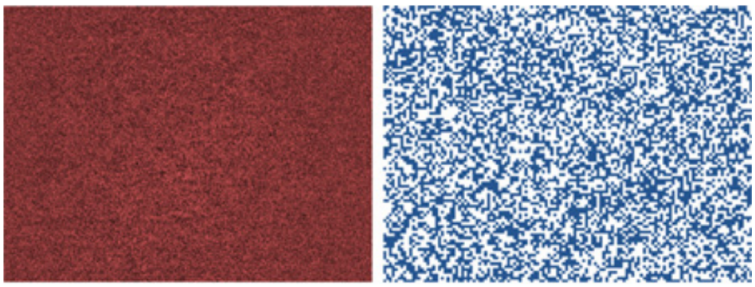

Figure 2.19: Comparison of the input image (left) and a created bitmap (right) of the zero/one output from the proposed TRNG algorithm of Zhang et al., 2012

the generation and usage of a one time pad, thus a key or phrase that is only used once. For the creation of the one time pad his proposal was to take the difference out of two independent pictures and use the blue color channel of those pictures. Only one color channel is selected to erase correlations between the other color channels that may exist. As part of the generation process an XOR operation in correlation with RFC4086 from Schiller, 2005, is done on the input pictures. After the mixing some pixels are dumped to minimize or even extinguish possible relations of consecutive pixels (Leschiutta, 2015), (Leschiutta, 2016).

Figure 2.20: Used input image 'a' (Leschiutta, 2016)

Figure 2.21: Used input image 'b' (Leschiutta, 2016)

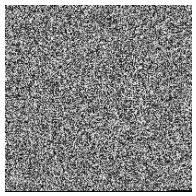

Figure 2.22: Resulting bitmap output of 'a' and 'b'(Leschiutta, 2016)

Moreover, the proposed and provided BlueRand Java Application from Leschiutta, 2016, provides not only a generation of random numbers out of two pictures but also a multi picture mode. If the proposed algorithm is fed not only by two but by a lot of other pictures even more randomness can be extracted. The results of two test suites, ENT, and RaBiGeTe were interesting according to Leschiutta, 2016. Unfortunately, no results of the NIST Statistical Test Suit were available for a comparison with the Master Thesis at hand.

2.7.3 Proposed RNG by Chen, 2013

The research article by Chen, 2013, is about extracting randomness out of video and audio sources. His proposed algorithm is on a bitwise operation on the base color chan-

nels Red, Green and Blue and in correlation to the audio within that specific coordinate (x,y) on the image. His proposed algorithm shown in Figure 2.23, shows that the RNG works in different stages and has a security threshold implemented, which drops the frame once it is triggered. This security threshold is reached when the counter variable 'watchdog' is larger than the predefined threshold 'th'. 'W' and 'H' in his proposed algorithm stands for the width and height of the input picture and is used for modulo operation in the last part of the randomness extraction. In his conclusion he pointed out that it is possible to generate strong random numbers without the usage of expensive equipment. The output of the proposed RNG was tested with the NIST Statistical Test Suite and passed all 15 tests (Chen, 2013).

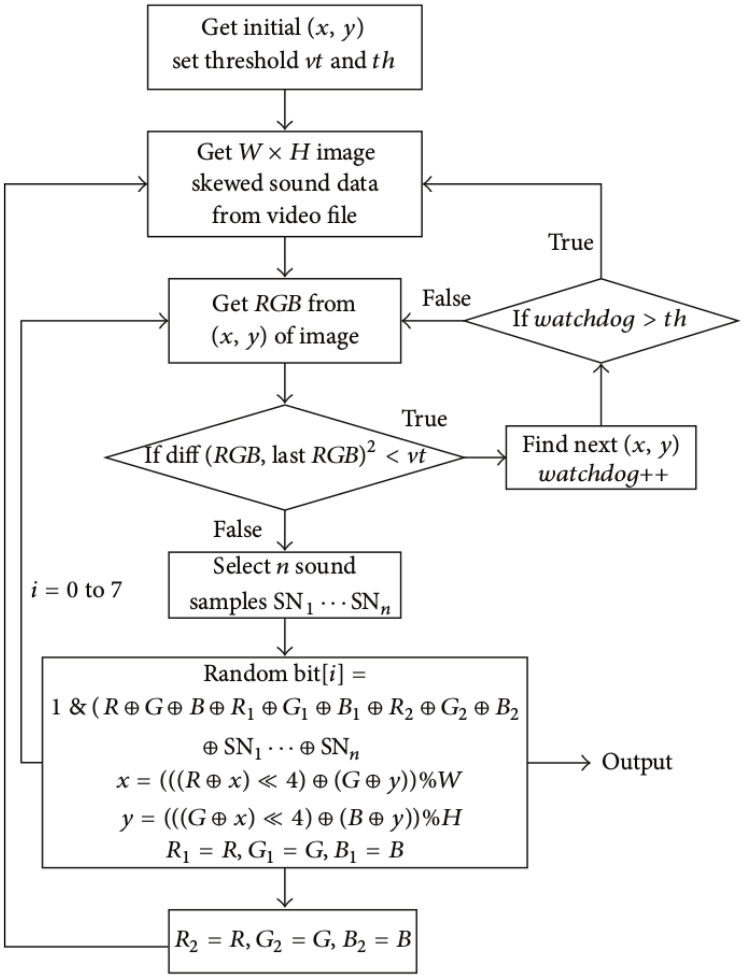

Figure 2.23: Showing the flowchart of the proposed algorithm of Chen, 2013, for a Random Number Generator using audio and video sources

2.7.4 Proposed RNG by Krhovják, Švenda, and Matyáš, 2007

In a paper published by Krhovják, Švenda, and Matyáš, 2007, on Masaryk University in Brno the authors of the paper examined different random sources available on smartphones, not only by their availability but also on their quality. On their analysis they focused on the camera and the microphone. As testobjects the smartphones Nokia N73, E-Ten X500 and E-Ten M700 devices were used for their practical experiments. As the paper points out, issues with API restrictions appered (Krhovják, Švenda, and Matyáš, 2007).

According to Krhovják, Švenda, and Matyáš, 2007, uncertainty or entropy is a basic measurement in information theory and often written as formula as:

$$H_1(X) = -\sum_{x \in X} P_X(x) log P_X(x) \tag{2.2}$$

They also point out that this formula is often referenced as Shannon Entropy due to the fact that this formula is about the average entropy. On the other hand, there is a min-entropy formula that measures the worst case entropy. It is defined as:

$$H_\infty(X) = min_{x \in X}(-log P_X(x)) = -log(max_{x \in X} P_X(x)) \tag{2.3}$$

Here the sample x is drawn from random distribution X with the probability $P_X(x)$. According to the sources of Krhovják, Švenda, and Matyáš, 2007, those two entropy measures are special cases of the generalised so called Rényi entropy. Random bits were gathered in four different ways: processing raw values only, least significant color bit, color combination using XOR and Flip-Flop bit extraction. Where the first three methods failed, the most robust one was the Flip-Flop bit extraction. By Flip-Flop extraction the authors meant processing each pixel $\pmod 2$. The result of each Flip-Flop was concatenated to the end of the gathered bit stream. But also with this method not all tests of the NIST Statistical Test Suite passed. The test suite was fed with '100 sequences per 1Mb' (Krhovják, Švenda, and Matyáš, 2007).

Unfortunately, the description of how the test suite was fed is not clear as it could mean that 1Mb = 1.000.000 bits and was equally distributed on each sequence/bit stream, thus 100 sequences with 1000 bits or 100 sequences with each containing 1.000.000 bits.

2.8 Purpose of Statistical Tests

In this section, the 15 different tests from the test suite provided by the National Institute of Standard and Technology (NIST) are discussed. The tests were developed to examine randomness of binary sequences produced by any Random Number Generator and focused on a variety of different types of non-randomness that could exist in a sequence (Rukhin et al., 2010, p.2-1). Some of the tests in this suite had the 'standard normal' and the 'chi-square(X^2)' as reference distribution. The calculated test statistic would fall in extreme regions of the reference distribution, if the sequence under test is in fact non-random. For all 15 tests of the test suite listed below the so-called "decision rule" was that if the computed P-value fell below 0.01 then the tested sequence was non random.

The used parameters for the statistical tests are described in Chapter 3.4.

2.8.1 Mathematical Definitions

Symbol	Meaning
S_n	The n^{th} partial sum for values $X_i = -1, +1$; *i.e., the sum of the first n values of* X_i
s_{obs}	The observed value which is used as a statistic in the Frequency Test
M	The number of bits in s a substring (i.e. block) being tested
N	The number of M-bit blocks to be tested
$\nabla^2\psi_m^2(obs)$	A measure of how well the observed values matches the expected value
$\nabla\psi_m^2(obs)$	A measure of how well the observed values matches the expected value
X^2	The theoretical chi-square distribution
$X^2(obs)$	The chi-square statistic computed on the observed values

Table 2.2: Table of symbols and mathematical definitions used in this thesis but taken from Rukhin et al., 2010

2.8.2 The Frequency (Monobit) Test

The proportion of zeroes and ones for the entire sequence is the focus of this test. To determine whether the number of ones and zeros in a sequence are approximately the same, as would be expected for a truly random sequence, the closeness of the fraction of ones to $\frac{1}{2}$ is assessed by this test. On the passing of this test all subsequent tests depends on (Rukhin et al., 2010, p.2-2).

It has to be considered that by analysing the results that if the P-value were small (< 0.01), then this would be caused by $\mid S_n \mid$ or $\mid s_{obs} \mid$ being large. Large positive values of S_n are indicative of too many ones, and large negative values of S_n are indicative of too many zeros.

According to Rukhin et al., 2010, the size for each sequence to be tested should be equal, or above 100 bits (i.e. $n \geq 100$).

2.8.3 Frequency Test within a Block

The proportion of ones within M-bit blocks is the focus of this test, as it determines whether the frequency of ones in an M-bit block is approximately $\frac{M}{2}$, as it would be expected under an assumption of randomness. For block size M=1, this test degenerates to test in Section 2.8.2, the Frequency (Monobit) Test (Rukhin et al., 2010, p.2-4).

According to Rukhin et al., 2010, it would have indicated that small P-values (<100) have a large deviation from equal proportions of zeros and ones in at least one the blocks.

According to Rukhin et al., 2010, the size for each sequence to be tested should be equal, or above 100 bits (i.e. $n \geq 100$). Note that $n \geq MN$. The block size M should be selected such that $M \geq 20, > .01n$ and $N < 100$.

2.8.4 The Runs Test

The focus of this test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits. A run of length k consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. The purpose of The Runs Test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow (Rukhin et al., 2010, p.2-5).

An oscillation in the string which is too fast would be indicated by a large value for $V_n(obs)$; too slow oscillation would be indicated by a small value. (A change from a one to a zero or vice versa is considered to be an oscillation.) When there are many changes a fast oscillation occurs, e.g., 010101010 oscillates with every bit. A slow oscillation stream has fewer runs than expected in a random sequence e.g., a sequence

containing 100 ones, followed by 73 zeroes, followed by 127 ones (a total of 300 bits) would only have three runs but 150 runs would be expected.

According to Rukhin et al., 2010, the size for each sequence to be tested should be equal, or above 100 bits (i.e. $n \geq 100$).

2.8.5 Tests for the Longest-Run-of-Ones in a Block

The longest run of ones within M-bit blocks is the focus of this test to determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones which would be expected in a random sequence. If an irregularity in the expected length of the longest run of ones occurs it implies that there is also an irregularity in the expected length of the longest run of zeroes which makes a test only for ones necessary (Rukhin et al., 2010, p.2-7).

It indicates that a tested sequence has clusters of ones when $x^2(obs)$ has large values.

The length of each block. The test code has been pre-set to accommodate three values for M:$M = 8$, M:$M = 128$, M:$M = 10^4$, in accordance with the following values of sequence length, n:

Minimum n	M
128	8
6272	128
750000	10^4

Table 2.3: Input size (n) recommendation for Longest-Runs-Of-Ones Test

2.8.6 The Binary Matrix Rank Test

The rank of disjoint sub-matrices of the entire sequence is the focus of this test to check for linear dependence among fixed length substrings of the original sequence. This test also appears in the *DIEHARD* battery of tests (Rukhin et al., 2010, p.2-10).

A deviation of the rank distribution from that corresponding to a random sequence would be indicated by large values of $x^2(obs)$ (and hence, small P-values).

M = Q = 32 and their probabilities have been calculated and inserted into the test code. If other choices than M and Q may be selected the probabilities would need to be calculated. The minimum number of bits to be tested must be such that $n \geq 38MQ$

(i.e., at least 38 matrices are created). For $M = Q = 32$, each sequence to be tested should consist of a minimum of 38,912 bits.

2.8.7 The Discrete Fourier Transform (Spectral) Test

The peak heights in the Discrete Fourier Transform of the sequence is the focus of this test to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence as it would indicate a deviation from the assumption of randomness. It is intended to detect whether the number of peaks exceeding the 95 % threshold is significantly different than 5 % (Rukhin et al., 2010, p.2-12).

Too few peaks (< 95%) below T and too many peaks (more than 5%) above T are indicated by a d value that is too low.

According to Rukhin et al., 2010, the size for each sequence to be tested should be equal, or above 1000 bits (i.e. $n \geq 1000$).

2.8.8 The Non-overlapping Template Matching Test

The number of occurrences of pre-specified target strings is the focus of this test to detect generators that produce too many occurrences of a given non-periodic (aperiodic) pattern. An m-bit window is used to search for a specific m-bit pattern and for the Overlapping Template Matching Test of Section 2.8.9. The window slides one bit position if the pattern is not found. If the pattern is found, the window is reset to the bit after the found pattern, then the search resumes (Rukhin et al., 2010, p.2-14).

The sequence has irregular occurrences of the possible template patterns if the P-value is very small (< 0.01).

Templates for $m = 2, 3, ..., 10$ are provided by the test code. To obtain meaningful results it is recommended that $m = 9$ or $m = 10$. The code may be altered to other sizes although $N = 8$ has been specified in the test code. To be assured that the P-values are valid N should be chosen such that $N \leq 100$. Also be sure that $M > 0.01 * nandN = \lfloor n/M \rfloor$.

2.8.9 The Overlapping Template Matching Test

The number of occurrences of pre-specified target strings is tested by the Overlapping Template Matching test. An m-bit window to search for a specific m-bit pattern is used for this test as well as the Non-overlapping Template Matching Test of Section 2.8.8. The window slides one bit position if the pattern is not found, as with the test in Section 2.8.8. This test and the test in Section 2.8.8 differ in such way that when the pattern is found, the window slides only one bit before resuming the search (Rukhin et al., 2010, p.2-17).

If the entire sequence had too many 2-bit runs of ones for the 2-bit template (B = 11), then firstly v_5 would have been too large, secondly the test statistic would be too large, and lastly the P-value would have been small (< 0.01) and 4) a conclusion of non-randomness would have resulted.

Each sequence to be tested consists of a minimum of 10^6 bits (i.e., $n \geq 10^6$) in regards to the chosen values of K, M and N.

2.8.10 Maurer's "Universal Statistical" Test

The number of bits between matching patterns (a measure that is related to the length of a compressed sequence) is the focus of this test to detect whether or not the sequence can be significantly compressed without loss of information. An occurrence of a significantly compressible sequence is considered to be non-random (Rukhin et al., 2010, p.2-20).

2.8.11 The Linear Complexity Test

The length of a linear feedback shift register (LFSR) is the focus of this test to determine whether or not the sequence is complex enough to be considered random. Longer LFSRs characterize random sequences. A too short LFSR implies non-randomness (Rukhin et al., 2010, p.2-24).

A P-value < 0.01 would have indicated that the observed frequency counts of T_i stored in the $v_I$bins varied from the expected values. The distribution of the frequency of the T_i (in the $v_I$bins) is expected to be proportional to the computed π_i.

Choose $n \geq 10^6$. For the results to be valid the value of M must be in the range $500 \geq M \geq 5000$, and $N \geq 200$ for the x^2.

2.8.12 The Serial Test

The frequency of all possible overlapping m-bit patterns across the entire sequence is the focus of this test. It has to be determined whether the number of occurrences of the 2^m-bit overlapping patterns is approximately the same Every m-bit pattern has the same chance of appearing as every other m-bit pattern which leads to random sequences having uniformity. Note that for m = 1, the Serial Test is equivalent to the Frequency Test of Section 2.8.2, (Rukhin et al., 2010, p.2-26).

If $\delta^2\psi_m^2$ or $\delta\psi_m^2$ had been large it is implied that non-uniformity of the m-bit blocks is given.

Choose m and n such that $m < \lfloor \log_2 n \rfloor - 2$.

2.8.13 The Approximate Entropy Test

The focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence as with the Serial Test of Section 2.8.12. Comparing the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a random sequence is the purpose of this test (Rukhin et al., 2010, p.2-29).

Strong regularity would be implied by small values of $ApEn(m)$. A substantial fluctuation or irregularity would be implied by large values.

Choose m and n such that $m < \lfloor \log_2 n \rfloor - 5$.

2.8.14 The Cumulative Sums (Cusums) Test

The maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence is the focus of this test. Whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences shall be determined by this test. A random walk may be considered by this cumulative sum. The excursions of this random walk should be near zero for a random sequence, for certain types of non-random sequences the excursions of this random walk from zero will be large (Rukhin et al., 2010, p.2-31).

Large values of the statistic mode = 0 indicate that there are either “too many ones” or “too many zeros” at the early stages of the sequence. Large values of the statistic

mode = 1 indicate that there are either “too many ones” or “too many zeros” at the late stages. A too evenly intermixing of ones and zeros would be indicated by small values of the statistic.

According to Rukhin et al., 2010, the size for each sequence to be tested should be equal, or above 100 bits (i.e. $n \geq 100$).

2.8.15 The Random Excursions Test, and

The number of cycles having exactly K visits in a cumulative sum random walk is the focus of this test. After the (0,1) sequence is transferred to the appropriate (-1, +1) sequence the cumulative sum random walk is derived from partial sums. A sequence of steps of unit length taken at random that begin at and return to the origin is what a cycle of a random walk consists of. The determination if the number of visits to a particular state within a cycle deviates from what one would expect for a random sequence is the purpose of this test. This test is a series of eight tests (and conclusions) with one test and conclusion for each of the states: $-4,-3,-2,-1$ *and* $+1,+2,+3,+4$ (Rukhin et al., 2010, p.2-33).

If x^2(obs) were too large the sequence would have displayed a deviation from the theoretical distribution for a given state across all cycles.

According to Rukhin et al., 2010, the size for each sequence to be tested should be equal, or above 1,000,000 bits (i.e. $n \geq 10^6$).

2.8.16 The Random Excursions Variant Test

The total number of times that a particular state is visited (i.e., occurs) in a cumulative sum random walk is the focus of this test so that deviations from the expected number of visits to various states in the random walk can be detected. This test is a series of eighteen tests (and conclusions), one test and conclusion for each of the states: -9, -8, ..., -1 and +1, +2, ..., +9 (Rukhin et al., 2010, p.2-38).

According to Rukhin et al., 2010, the size for each sequence to be tested should be equal, or above 1,000,000 bits (i.e. $n \geq 10^6$).

Chapter 3

Implementation

In the pre-research to this Master Thesis there was the idea of extracting randomness by collecting information about occurring random bitflips when cosmic rays or highly energetic particles like muons (see Chapter 2.5) hit the silicate built-in storage components of consumer electronics. However, this attempt failed due to several reasons in the execution, therefore the strong results can not be verified. In the end this experiment was very educational and even though the results of this experiment can not be used, existing literature proofs that this approach, of using muons to extract randomness, is theoretically possible. The implementation of the Economy Random Bit Generator (ERBG) can be found in Chapter 3.1. Nevertheless, the experiment of using cosmic rays as entropy source is included at the very end of this chapter.

3.1 Practical Implementation of ERBG as Android Application

This section describes the implementation of the ERBG as an Android application. For the development the Android Studio provided by Google Inc. was used. As within the proposed algorithm in Listing 3.4, can be seen, the main operation that is executed on every byte is $\pmod 2$. This technique was also used in other existing RNGs but always limited to the color channel. In this implementation on each byte a $\pmod 2$ operation took place. Thus, it was distinguished between a random zero and one bit by using the Equation 3.1 and 3.2 as also shown below:

$$Zerobit = 0 \equiv byte \pmod 2 \tag{3.1}$$

$$Onebit = 1 \equiv byte \pmod 2 \tag{3.2}$$

This operation allows to break every byte down into two possible options: zero or one. As described in Chapter 3.5.2, the decision was made to cut the first 4000 bits due to a permanent occurrence of approximately 3250 zero bits at the beginning of each file. This solution solved also the "igamc: UNDERFLOW" error as described in Chapter 3.5.3.

3.1.1 Development

The development for the Android application of ERBG as well as the Java Standalone version of it was done on a Linux operating system with the usage of Android Studio. The full specifications are listed in Table 3.1.

Setting	**Value**
Operating System	
Operating System Publisher	Zorin 12.3
Operating System Kernel	#31~16.04.1-Ubuntu SMP Wed Jul 18 08:54:04 UTC 2018 4.15.0-29-generic
Integrated Development Environment (IDE)	
Android Studio	Android Studio 3.1.1
Build	AI-173.4697961, built on April 4, 2018
JRE	1.8.0_152-release-1024-b01 amd64
JVM	OpenJDK 64-Bit Server VM by JetBrains s.r.o

Table 3.1: Detailed specification of the used development environment

For the Android application of ERBG the Model-View-Control pattern, as can be seen in Figure 3.1, was used. At the startup of the application the *MainActivity* was called to prepare the buttons of the different entropy sources for a proper usage. Notice that the *Picture Mode* is not implemented and is a placeholder for further research after this Master Thesis at hand. As the application was developed with so-called fragments, it is possible to press the back button for returning to the previous screen. Each fragment holds the implementation for the assigned entropy source, for instance, by pressing the *Video Mode* button within the main screen shown in Figure 3.2, the *VideoFragment* is called. The *VideoFragment* screen, as shown in Figure 3.3, is capable of recording a video, thus gathering a raw material video. During an ongoing recording the button

Record is renamed to *Stop* for finishing the capturing process. After a raw material video is gathered, the *RandomService* service is called. The *RandomService* is responsible for providing methods such as generating correct file paths, session identification as well as the randomness extraction process with basic statistics about the distribution of zeros and ones. However, after the *RandomService* has finished the extraction of randomness the results of the basic statistics are handed over to the *Result Fragment* which is implemented in the *ERNGFragment* class. Note that during the whole random extraction process the resulting file is written constantly, thus incremental. This allows to prevent a possible loss of data in case something unexpected is happening but also allows to capture a file without the limitation of local variables.

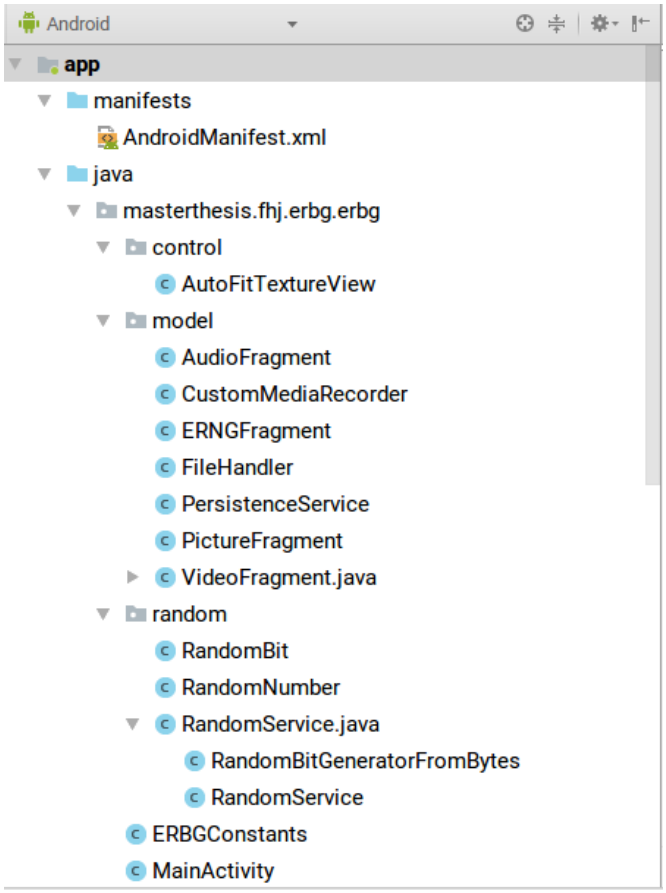

Figure 3.1: Shows the structure of packages and classes for the source code of ERBG

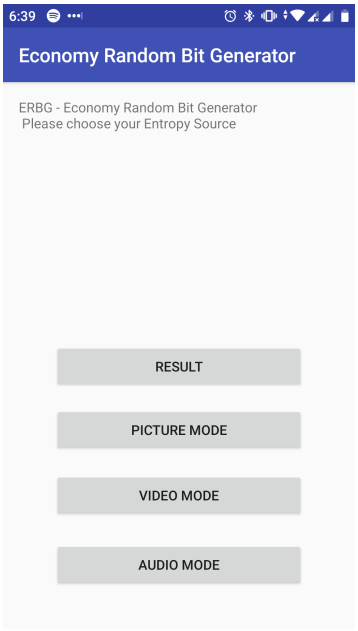

Figure 3.2: A screenshot of the main view within the ERBG Android application

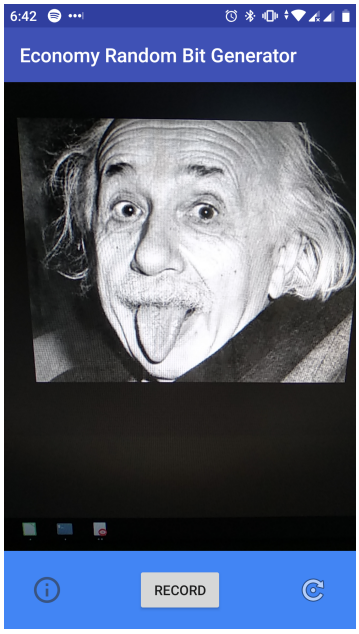

Figure 3.3: A screenshot of the video view within the ERBG Android application

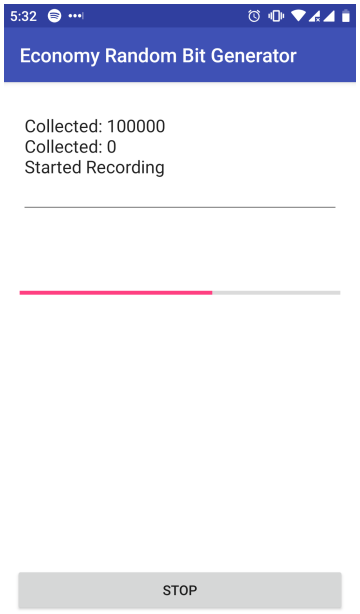

Figure 3.4: A screenshot of the audio view within the ERBG Android application

3.2 Homemade Tools and Helpers

In this section tools and very supportive commandline statements are described. Those little helpers were developed while completing this Master Thesis. As those gadgets enormously supported the progress of this research and simplified many things they do deserve to be taken into account.

3.2.1 EBSO

EBSO is the acronym and stands for **E**conomy Random Bit Generator - **B**asic **S**tatistic **O**utput. This tool is written as bash script, which can be seen in Listing 3.2.1, where the usage output printed onto the terminal. Due to the fact that this bash script is deposited within "/usr/bin" as a symbolic link, it can be called throughout the system within any shell. To get files from the current working directory the variable "$PWD" was used within the script of EBSO. However it does not calculate the longest run of zeros or ones as both other implementations, the Android ERBG as well as the Java Standalone ERBG does.

```
1 ignis@Sequoia:/usr/bin\$ ls -lha  ebso
```

```
2 lrwxrwxrwx 1 root root 39 Mar 16 12:55 ebso -> /home/ignis/install/
3 erbg\_basic\_stats.sh
```

Listing 3.1: Position of Economy Random Bit Generator - Basic Statistic Output (EBSO) within the Linux file system

```
1 ignis@Sequoia:~\$ ebso
2 Economy Random Bit Generator - Basic Statistic Output (EBSO)
3 usage: ebso <filename>
4
5 Computes basic information about a given file containing one and zero
6 characters. This output is written to console as well as into a file
7 called erbg\_basic\_stats.txt
8
9 (c) Stefan Kutschera, BSc [dmz.stefan.kutschera@gmail.com]
```

Listing 3.2: Terminal output for the usage of Economy Random Bit Generator - Basic Statistic Output (EBSO)

3.2.2 ERBG - Java Standalone

As the Android application was tested on a few different smartphone devices it did not take long until someone tried to send a large file for analysis within the scope of this Master Thesis at hand. Due to the upload speed and the lack of patience caused by my personal excitement, the idea of processing the gathered raw material data was born. After a small amount of successful tests it was proofen that the ERBG Android application as well as the ERBG Java Standalone Application extracts, thus generates, the same output file out of the same raw material file. This fact is also important as it is now possible to test other file structures. Moreover, the ERBG Java Standalone Application is the encapsulated and easier understandable version due to the fact that the Android application implements a lot of necessary overhead to function on a smartphone. In addition it implements the gathering process.

The full source code for the ERBG Java Standalone Application can be found in the Appendix Chapter A.3, of this Master Thesis at hand. The results after executing the ERBG Java Standalone Application on the terminal are shown in Figure 3.5.

3.2.3 Usefull Commands and Tips and Tricks

During the Master Thesis at hand some minor issues occurred which were solvable at a first glance but took annoyingly long to process. Out of the nature of my studies I

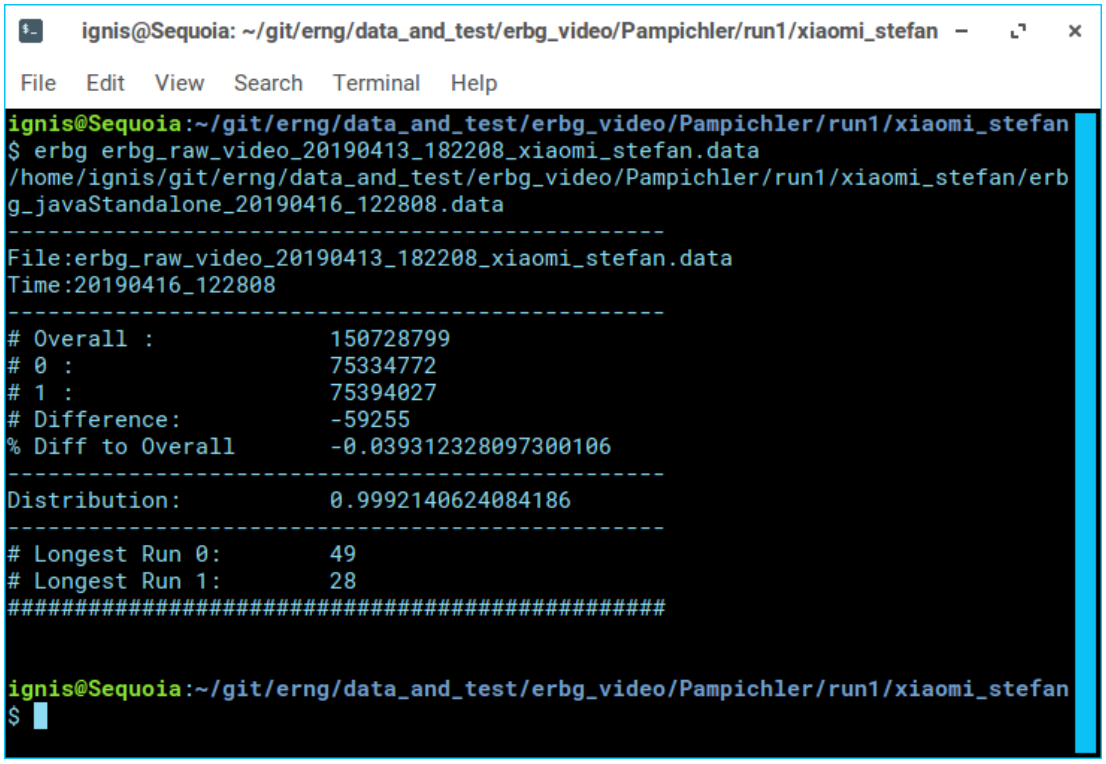

Figure 3.5: Shows very basic statistical information results after executing the ERBG Java Standalone Application on the terminal

use the force of computational systems to ease and simplyfiy action sequences. For instance, the output of the QRNG Quantis from ID Quantique, Swiss Quantum, 2016, produced a random sequence where each character was separated by a comma, for example '0,1,1,0,1,0,1,0,0,'. Unfortunately the NIST Statistical Test Suite was not happy about such input. A search and replace forced me to actually open the file, which took some time and was somewhat unproductive as well as repetitive. As shown in Listing 3.3, line two, the simple commandline allowed me not only to get the same result within less time, it also enabled the removal of newlines as shown in the same figure in line five. Henceforth, I used and adapted the list of useful commands, which were very helpful to me and hopefully anyone else who will read this Master Thesis.

```
1 #Clean the output from Quantis (remove all commas)
2 sed -i 's/,//g' file.dat
3
4 #remove all newlines in
5 sed ':a;N;$!ba;s/\n/ /g' file.dat
6
7 #add newline after every 20 char
8 sed -e "s/.\{20\}/&\n/g" < temp.txt
9
10 #Generate a random stream from /dev/urandom
```

```
11 cat /dev/urandom | tr -dc '0-1' | fold -w 102400 | head -n 1
12
13 # Count the 'word' 1 within the file 'somefile.txt'
14 grep -o -i 1 someFile.txt | wc -l
15
16 # within the NIST Statistical Test Suite it was able to clear maybe
17 # existing previous results and start the test suite afterwards
18 cd experiments/ && ./cleanup_results.sh && cd .. && ./assess 1000000
```

Listing 3.3: Shows a collection of used and very helpful commands

3.3 Proposed Algorithm

Within this section the used and proposed algorithm for the implementation of ERBG are discussed. Whereas the algorithm of gathering randomness from the video entropy source was build in within the ERBG Java Standalone Application, the proposed algorithm of gathering randomness from the audio entropy source can only be found within the Android implementation. However, both proposed algorithm are using the same methods for extracting randomness.

3.3.1 Video Algorithm

In Listing 3.4, the used algorithm is shown that is used in both, the video extraction mode of the ERBG Android application as well as in the ERBG Java Standalone Application. The listing also includes the implementation of the calculation the count of zeros and ones as well as a count of the longest sequence of identical bits, thus the longest run of ones or zeros.

```
1 /**
2  * This is more or less the heart of the ERBG. It extracts zero or
3  * one by simply calculating 'byte modulo(2)' from any given byte
4  * array. This is normally called by the method 'generateFromFile()'
5  * @param input
6  */
7 private void byteArray2binary(byte[] input){
8   Log.i(ERBGConstants.LOG_TAG_ASYNC,
9     "There are " +input.length +" Bytes to process");
10  StringBuilder sb = new StringBuilder();
11  for (byte b:input) {
12    if (firstBytesDumpCounter > ERBGConstants.FIRST_BYTE_DUMP_THRESHOLD) {
```

```
13        int i = Math.abs(b)%2;
14        if (i==0) {
15          zero++;
16          actRunZero++;
17          actRunOne = 0;
18          if(actRunZero>longestRunZero) {
19            longestRunZero = actRunZero;
20            }
21          }
22        else {
23          one++;
24          actRunOne++;
25          actRunZero = 0;
26          if(actRunOne>longestRunOne) {
27            longestRunOne = actRunOne;
28          }
29        }
30      sb.append(i);
31      }
32      firstBytesDumpCounter++;
33    }
34    sb.append("\n");
35    writeToFileExternal(sb);
36    }
```

Listing 3.4: Main function including the algorithm that extracts the randomness by executing the modulo 2 operation

3.3.2 Audio Algorithm

Where as the proposed algorithm for the used audio entropy source, as shown in Listing 3.5 has gathering and extraction implemented at once but lacks in the counting of zeros and ones as well as the longest runs of those.

```
1   /**
2    * Method for collecting audio data AND extract randomness
3    */
4   private void collectAudioData() {
5     // Write the output audio in byte
6     short sData[] = new short[BufferElements2Rec];
7     int count = 0;
8     StringBuilder sb = new StringBuilder();
9     log("Started Recording");
10    while (isRecording) {
```

```
11        long amp = 0;
12
13        // gets the sound input from microphone as byte format
14        int readsize = recorder.read(sData, 0, BufferElements2Rec);
15        long sum = 0;
16        for (short data : sData) {
17          sum += data;
18          if (count >= 200)
19            sb.append(Math.abs(data) % 2);
20          count++;
21          if (((count - 200) % 100000) == 0) {
22            log("Collected: " + (count - 200));
23            this.writeToFileExternal(sb);
24            sb = new StringBuilder();
25          }
26        }
27        if (sum <= 0)
28          amp = (sum * (-1)) % 100;
29        mProgressBar.setProgress((int) amp);
30      }
31      log("Finished Recording");
32    }
```

Listing 3.5: Shows the used algorithm for gathering and extracting randomnes from an audio entropy source

3.4 Test Parameters

As some parameters are already defined and described in Chapter 3.5.1, it is easier to reproduce if the used values are in a clear and consistent format. To be competitive with other existing literature the length for a bit stream is $1x10^6$, as this value is a common subset of possible values for each of the 15 statistical tests within the NIST Statistical Test Suite provided by Rukhin et al., 2010.

3.5 Problems

During every scientific work there are problems to solve, this Master Thesis is not an exception to that. Therefore, in the following section the problems that occurred during this thesis will be discussed as well as their solution.

Parameter	Value
Length of a bitstream	$1x10^6$
Amount of bitstreams	100
Applied statistical tests	1 - 15
Input file format	ASCII
Block Frequency Test - block length(M)	128
NonOverlapping Template Test - block length(m)	9
Overlapping Template Test - block length(m)	9
Approximate Entropy Test - block length(m)	10
Serial Test - block length(m)	16
Linear Complexity Test - block length(M)	500

Table 3.2: Used parameters and settings to run the NIST Statistical Test Suite

3.5.1 Run NIST Statistical Test Suite

The following problem may be not obvious, because it is well documented how to get the Test Suite running. In the very early stages of research I tried to get the NIST Statistical Test Suite up an running with the output of the QRNG Quantis from IDQuantique. The output of this fine art of technology was a bad formatted file with zeros and ones separated by commas. The other more difficult issue was the misconfiguration and misuse of the test suite.

3.5.1.1 Solution

In order to get proper results out of the Statistical Test Suite provided by Rukhin et al., 2010, the execution and configuration has to follow certain rules. It is necessary to include it to avoid some peculiar errors I found myself running into, which were guilted to misconfiguration but also misleading literature about certain errors. After re-analysing the misleading literature I found out that the problem may be solved by using the provided test suite in a proper way. Nonetheless, a successful run of a tested file with the output log can be seen in Listing 3.7.

In order to start and run the Statistical Test Suite it has to be downloaded from the NIST website [1]. When it comes to starting the tests, the executable binary 'assess' lies within the folder of the Statistical Test Suite as shown in Figure 3.6. As Rukhin et al., 2010, pp. 5-5, described in the documentation for the test suite, within the Section 5.6, the parameter needed by the test suite is the desired length of the bitstream n, thus

[1] https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software

how many bits a single bitstream should contain. Note that if the length of a bitstream multiplied by the amount of desired bitstreams is beyond the amount of containing bits within the tested file the test suite will not only run in several severe errors but will also most likely cause the current run to crash. In the documentation of Rukhin et al., 2010, a recommendation for the length of the bitstream n is described within in each statistical test.

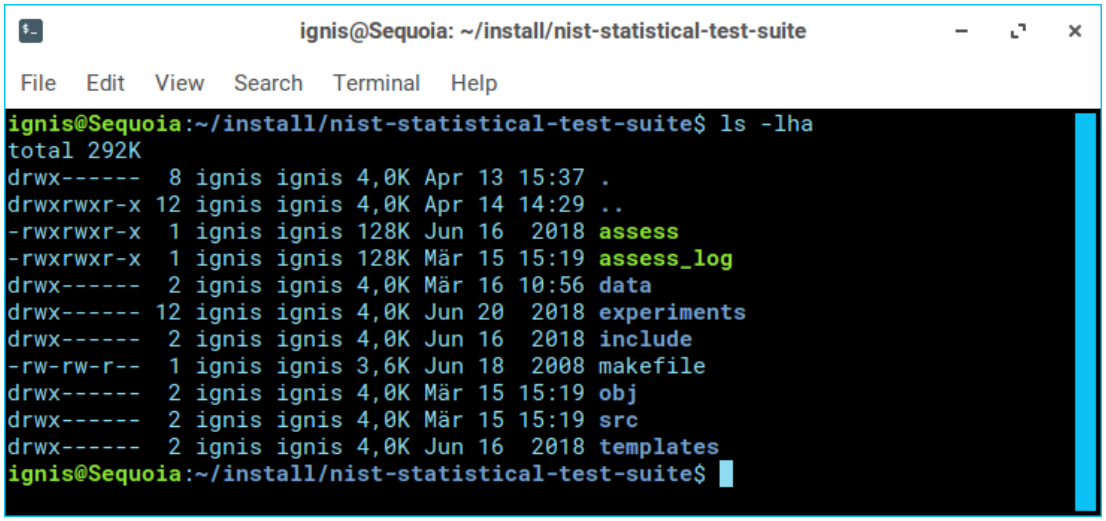

Figure 3.6: Terminal output of the listing command within the provided Statistical Test Suite from NIST

After selecting the desired "*Generator Selection*" which is a file stored locally, the option "[0] Input File" is chosen. Then the test suite needs an absolute path to the file under test. It can be very tricky to define which tests should be applied if its only necessary to take a few tests. In Figure 3.7, an example of the question matrix is listed. It has to be read from top to bottom starting in line seven to line nine of Listing 3.6. In the example listing, test number two and test number three will not be executed as their option is set to zero, all other tests will be executed.

```
1    Enter Choice: 0
2
3    INSTRUCTIONS
4    Enter a 0 or 1 to indicate whether or not the numbered statistical
5    test should be applied to each sequence.
6
7    123456789111111
8             012345
9    100111111111111
```

Listing 3.6: Selection Matrix on the applicable tests applied to the file under test

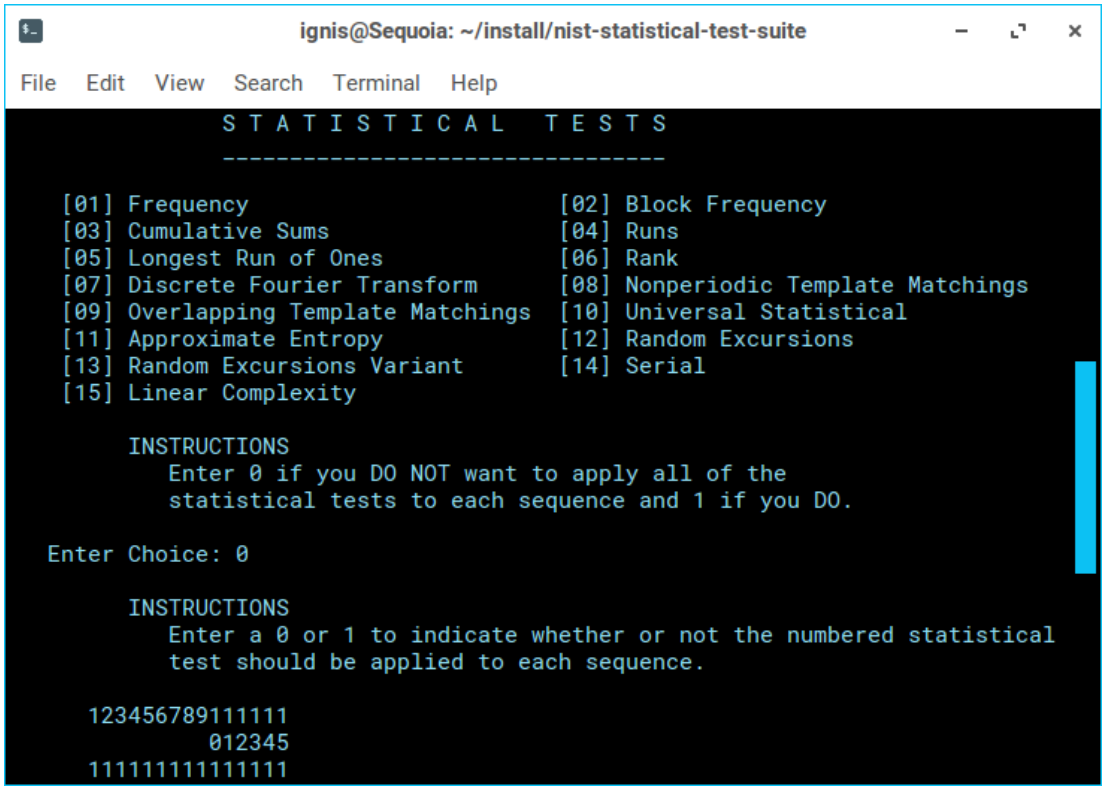

Figure 3.7: Terminal output of the test suite awaiting input for the chosen tests to be applied on the file under test

In one of the options following there is "*How many bitstreams?*". It is important to understand that the length of the bitstream n is already defined by the input parameter of the executed command "*./assess 1000000*". By choosing 100 as desired value for the tested file it must contain more or equal to $100 * 1000000$, thus $1x10^8$, bits.

```
ignis@Sequoia:~/install/nist-statistical-test-suite$ ./assess 1000000
          G E N E R A T O R    S E L E C T I O N
          ---------------------------------------

    [0] Input File                 [1] Linear Congruential
    [2] Quadratic Congruential I   [3] Quadratic Congruential II
    [4] Cubic Congruential         [5] XOR
    [6] Modular Exponentiation     [7] Blum-Blum-Shub
    [8] Micali-Schnorr             [9] G Using SHA-1

   Enter Choice: 0


               User Prescribed Input File: /home/ignis/install/
               nist-statistical-test-suite/data/erbg.data

```

```
                S T A T I S T I C A L   T E S T S
                _________________________________

 [01] Frequency                       [02] Block Frequency
 [03] Cumulative Sums                 [04] Runs
 [05] Longest Run of Ones             [06] Rank
 [07] Discrete Fourier Transform      [08] Nonperiodic Template Mat
chings
 [09] Overlapping Template Matchings  [10] Universal Statistical
 [11] Approximate Entropy             [12] Random Excursions
 [13] Random Excursions Variant       [14] Serial
 [15] Linear Complexity

      INSTRUCTIONS
         Enter 0 if you DO NOT want to apply all of the
         statistical tests to each sequence and 1 if you DO.

Enter Choice: 0

      INSTRUCTIONS
         Enter a 0 or 1 to indicate whether or not the numbered statistical
         test should be applied to each sequence.

   123456789111111
             012345
   111111111111111


     P a r a m e t e r   A d j u s t m e n t s
     -----------------------------------------
 [1] Block Frequency Test - block length(M):         128
 [2] NonOverlapping Template Test - block length(m): 9
 [3] Overlapping Template Test - block length(m):    9
 [4] Approximate Entropy Test - block length(m):     10
 [5] Serial Test - block length(m):                  16
 [6] Linear Complexity Test - block length(M):       500

Select Test (0 to continue): 0

How many bitstreams? 100

Input File Format:
 [0] ASCII - A sequence of ASCII 0's and 1's
 [1] Binary - Each byte in data file contains 8 bits of data
```

```
62    Select input mode:  0
63
64      Statistical Testing In Progress.........
65
66      Statistical Testing Complete!!!!!!!!!!!!
```

Listing 3.7: Shows the standard output from terminal during a successful test

3.5.2 Odd Pattern after Random Extraction Process

When the first tests failed constantly no matter how many different ways were tried to use a varying environment, a closer look at the extracted random file was taken. As it turned out, the first approximately 3250 bits had, with a few exceptions on the beginning, always just 0 as value as can be seen in Figure 3.8. However, when the raw file, thus the video file saved as mp4 container, was analysed there was no such obvious odd pattern recognizable, as can be seen in Figure 3.9. As a result, the statistical tests of the NIST Statistical Test Suite from Rukhin et al., 2010, did not always fail but threw an "igamc: UNDERFLOW" error when executed 3.4.

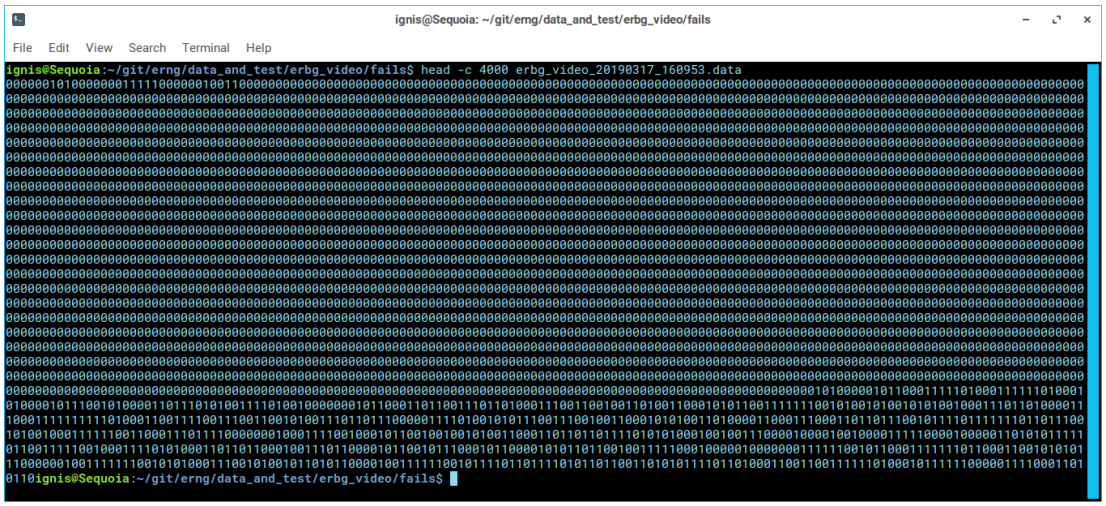

Figure 3.8: Terminal output of the first 4000 characters from the extracted random data file. It is clearly visible that there are lot of zeros at the beginning.

3.5.2.1 Solution

As a result, the first 4000 bits of each input data are not used for the extraction of randomness. As can be seen in line 13 of Listing 3.4, a byte counter is checked against the defined threshold before the current byte is being processed. Until the threshold is

Figure 3.9: Terminal output of the first 4000 characters from the raw material file that is later used to extract randomness. When this file is used to extract randomness, the first 3250 characters are most likely to be zeros.

reached every byte is dumped and therefore not included within the final extracted random data. As an improvement of this solution the byte counter could be implemented outside the for-loop.

3.5.3 Error: “igamc: UNDERFLOW”

In some executions of the very first runs of the NIST Statistical Test Suite the information/error “igamc: UNDERFLOW” was thrown as can be seen in line 3-5 of Listing 3.8. Sometimes this error did not only occur a few times during one execution of the Statistical Test Suite, it occurred massively.

3.5.3.1 Solution

After doing some research, the book[2] of Johnston, 2018, led me in a completely wrong direction as his work is based on incorrect usage of the NIST Statistical Test Suite from Rukhin et al., 2010. It stated that the “igamc: UNDERFLOW” error was due to a wrong implementation of Rukhin et al., 2010. As it turned out later, Johnston, 2018, used ’./assess 1’ to start the Statistical Test Suite which simply means, that the given tested string is one integer long (see Chapter 3.5.1 and Chapter 3.4). The option for bitstreams was also chosen to be one by Johnston, 2018. As a result, the configuration of Johnston, 2018, for the NIST Statistical Test Suite tested one bitstream with a length of one bit.

After the creation of another very simple random number generator based on java.random, which just created zeros and ones, the tests did not fail. However, as it later turned out

[2]I want to point out that I did not read all of the 422 pages work. Therefore, the rest of Johnston, 2018, book may be correct from a scientific point of view. But it has to be taken into account that the described usage is a very significant misuse of the test suite.

this was not a failure of the testsuite nor its implementation, hence it came from the quality in respect to randomness, of the input. As further investigation found out the result from the equation of the Incomplete Gamma Function got a result that was under a threshold defined as $(-1) * 7.09782712893383996732224 * 10^2$. The final hint that lead into this issue was given by simply feeding the Statistical Test Suite with data only containing zeros. After this approach it was clear that the failures were not caused by wrong characters but rather by a quality issue of the input data provided by the ERBG.

```
1      Statistical Testing In Progress.........
2
3 igamc: UNDERFLOW
4 igamc: UNDERFLOW
5 igamc: UNDERFLOW
6      Statistical Testing Complete!!!!!!!!!!!!
```

Listing 3.8: Output of the terminal showing the occurence of the “igamc: UNDERFLOW” error during the test execution of the NIST Statistical Test Suite from

3.5.4 ERBG fails statistical test when data is gathered on other devices

After achieving the passing of every 15 of the statistical tests from NIST Statistical Test Suite, the Android application was compiled and used on other devices too. The first impression of the basic statements which were printed after randomness was extracted from a raw file which was odd but not alarming. In contrast, when feeding the Statistical Test Suite with the file of extracted randomness the results were devastating. To further investigate into the issue a video was taken on the main development device, which had some positive results in the past, as well as on the newly tested device, which had the above mentioned devastating results. A direct comparison of pictures taken of the exact same scene and time shows that the main development device (i.e. Xiaomi Mi A1) has a lot more motion blur compared to the picture taken on the second test device (i.e. LG Nexus 5X) which can be seen in Figure 3.10 and Figure 3.11.

3.5.4.1 ~~Solution~~

After long and extensive research it was not possible to find an explanation for that behavior. As it turned out some specifications for capturing a video are set on the fly by either the image sensor or the operating system on the smartphone. This configurations were on the one hand the framerate, codec, codec profile or resolution. After they were

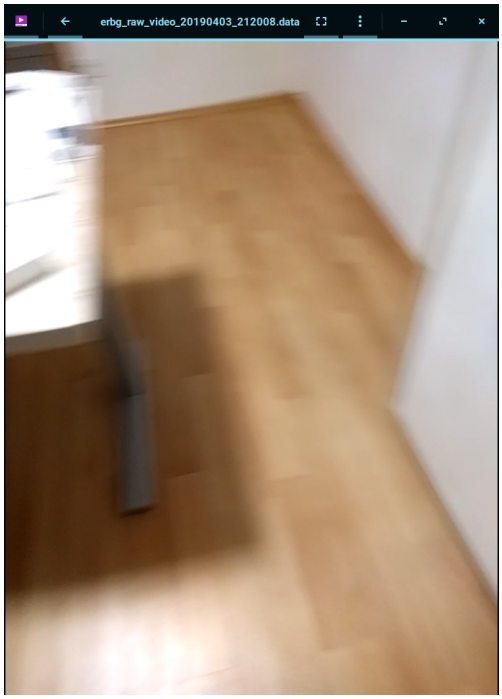

Figure 3.10: A snapshot of the exactly same scene, time and video that was also taken with another test device where the person who took the video was spinning in fast circles. Much less motion blur occurred than compared with the output of the main development device.

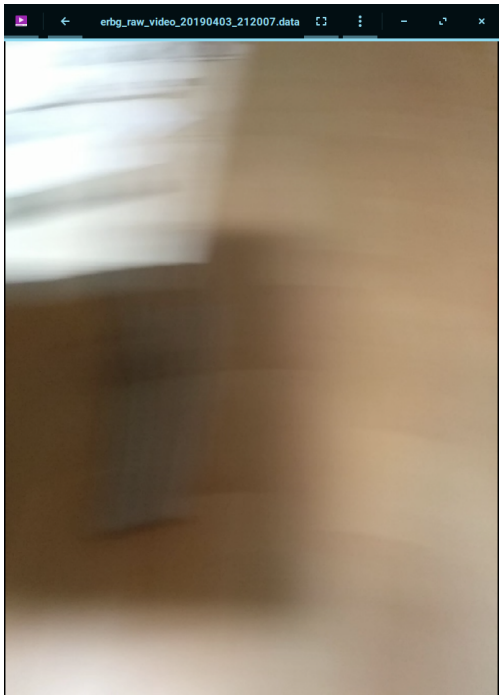

Figure 3.11: A snapshot of the exactly same scene, time and video that was also taken with another test device where the person who took the video was spinning in fast circles. Much less motion blur occurred than compared with the output of the main development device.

programmatically predefined to a fixed value, all gathered raw material videos had the same properties. However this did not fix this issue.
As the screenshot in Figure 3.10, looks like as it was stabilized from a software point of view, the next possible solution was to disable the *"Control Video Stabilization Mode"* and the *"Lens Optical Stabilization Mode"*. As it turned out, neither a separately or joined deactivation of the two stabilization modes was an improvement to the problem at hand.
As shown in Table 4.1, the image sensor type of the used primary camera of the device "LG Nexus 5X - Primary - Stefan" was a CMOS BSI, thus a backside illuminated image sensor, whereas the device "Xiaomi Mi A1 - Primary - Stefan" was based on a BSI basis but called "PureCel" by the manufacturer OmniVision. After some investigation it seemed that the image sensor type could have such impact, but using the secondary camera of the device "Xiaomi Mi A1 - Secondary - Stefan" the results were not comparable to the device "LG Nexus 5X - Primary - Stefan". Note that the device "Xiaomi Mi A1 - Secondary - Stefan" and the device "LG Nexus 5X - Primary - Stefan" have the same type of image sensors, thus the CMOS BSI, build in.
As a result, it cannot be said what caused the non-randomness on the device "LG Nexus 5X - Primary - Stefan" but it can be argued that neither the video settings, the image stabilization nor the image sensor type is the reason for that behavior.

3.5.5 Suddenly Unavailable Material

The literature as well as the implementation of the NIST Statistical Test Suite of Rukhin et al., 2010, are a very essential part of the Master Thesis at hand. Therefore, the loss of access to this material threatened the Master Thesis as a whole. Unfortunately, the time frame for my research was within the government shutdown of the United States of America. The shutdown took 35 days and happened between the 22th of December 2018 and the 25th of January 2019. As can be seen in Figure 3.12, access to proper material from the webservice of NIST was declined by an error message referring to the lack of governmental funding. Henceforth, every DOI-link[3] that led to the hosted webservice of NIST has shown the same error message, stating the service of NIST was unavailable due to the lack of governmental funding.

[3]Digital Object Identifier (DOI) https://www.doi.org/

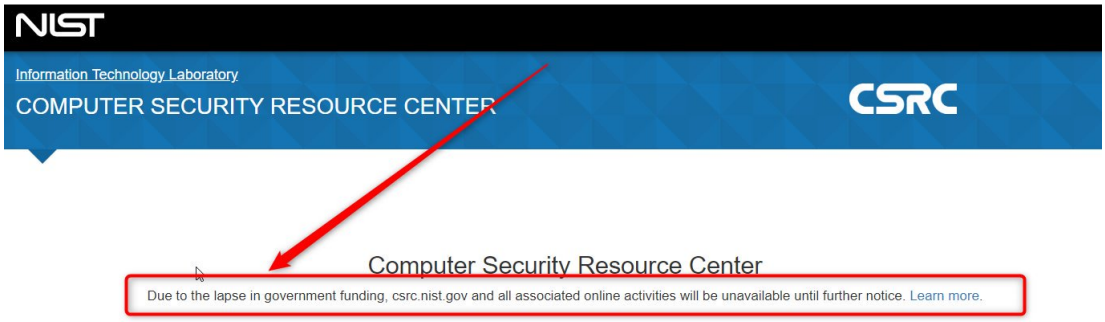

Figure 3.12: Error message of the NIST website while trying to download provided literature

3.5.5.1 Solution

The following is not a real solution, it is more a temporarily workaround. As the DOI-link https://doi.org/10.6028/NIST.SP.800-22r1a seems to correctly redirected the request to the desired URL, the information, as shown in Figure 3.12, of the governmental shutdown was shown. The visible URL was used and fed into the service of "*Internet Archive Wayback Machine*[4]". With that, I was able to get a version of the needed document from the 19th of December 2018[5], accordingly three days before the governmental shutdown started. The last published version of the needed document is from 2010 and has a version history within itself. This and the fact that every new released version supersedes the previous version would have allowed to take even a version of approximately two years ago.

The reason why this was just a temporarily solution is because there was no authenticity that the documents were unaltered and the same as provided from NIST. There was also no hash value given to prove if the available copy of the document was genuine.

3.6 Experiment: Using Cosmic Rays as Entropy Source

The aim of the experiment was to detect bitflips and measure their occurrence as well as their possible usage as an external source for generating random numbers. Even though the experiment itself failed due to two major errors in the experimental setup, as

[4] http://web.archive.org/
[5] http://web.archive.org/web/20181219061239/https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

well as the interference of the airport security, it is very important to include the failed experiment, to give others the possibility to learn from mistakes that were made. For the experiment different storage devices were overwritten with zeros from '/dev/zero'. If a highly energetic particle interferes with the storage medium and causes a bitflip it should be possible to see that some of the zeros have now become one.

3.6.1 Setup

Before the experiment, each device was completely overwritten from the output of '/dev/zero'. Therefore when reading the hard drive disc (HDD), the output of it was '0'. However, I failed by not re-reading all storage devices after the overwriting process. This is important because even though we can be highly certain that the writing process was a success, we cannot be definitely sure about the result.
Secondly, the probability of catching highly energetic particles is more common in higher altitudes. I took the storage devices on a trip to San Diego, California, U.S.A. The route was from airport Vienna via Munich to Los Angeles from where we traveled by train. The route is important due to the different altitudes on different flights as well as the sea-elevation level of the final destination.

- HDD 80GB as 'H80N1'
- USB Card violet 1GB as 'U01N1'
- USB Card white 4GB as 'U04N2'
- USB Stick silver 512MB as 'U01N3'
- SD Card black 256MB as 'C01N4'

3.6.2 Execution

For the experiment, each storage device was completely overwritten with zeros from '/dev/zero'. I did check storage device 'H80N1' after the initial zero-writing process but to my misfortune I did not check the other devices. In retrospective that was a huge mistake due to the uncertainty that the final results could derive from a failed or not completed zero-writing process. However, I took the storage devices in my carry-on baggage while travelling. What I did not consider in the beginning were the extensive security screenings at the airport. In fact, I was not prepared and had no other choice

as to let my carry-on baggage get scanned in the X-ray machines. Another mistake was not labeling all storage devices at the beginning of the experiment. This led to the effect that I was not sure which exact HDD I took with me on the trip as I had a number of similar HDDs at home. After the trip I read the data from the HDDs. As it is in their nature to produce heat during the reading process I used another HDD to protect my office table from taking damage due to the heat radiation of the HDD in the progress. During that I found myself in a situation were I was not able to distinguish the HDDs from each other anymore.
Nevertheless, I did not change the protocol as shown in Figure 3.13, due to the circumstance that I cannot be certain if the 'H80N1' was on the trip or not. That means that any results from 'H80N1' cannot be trusted.

3.6.3 Results of Experiment

Even though the experiment cannot prove that bitflips occurred due to exposure to cosmic rays or high energetic particles, it shows that a small amount of uncertainty bitflips occurred as it turned out that the devices 'U01N1' and 'U04N2' had much data on it which were not zero. In case of device 'U01N1' it is 0.276 GB of 1.0 GB of the storage capacity which is 26.7% of the overall capacity where as device 'U04N2' has 2.4 GB of 4.0 GB or 60% of non-zero values

Speed	EGREP Command
7 MB/s	sudo dd if=/dev/sda | xxd -p | egrep "[^0]+" > bitflip.log
3 MB/s	sudo dd if=/dev/sda | xxd | egrep "[1-9]{1,4}[0]{0,3} " > bitflip.log

Table 3.3: Speed comparison of the used non-zero check commands used on a linux command line

Process	Device Name	Device ID	Date/Time	Result	Additional Information					
			yyyy/mm/dd hh:mm		*Location*	*GPS longitude*	*GPS latitude*	*Elevation Flight altitude*	*Room properties*	
initial zero writing	HDD 80GB	H80N1	2018/08/25 05:00 PM GMT+2	no errors during process						
location change	HDD 80GB	H80N1	2018/08/25 06:00 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
check for bitflip	HDD 80GB	H80N1	2018/09/04 06:30 PM GMT+2	no bitflip found						
location change	HDD 80GB	H80N1	2018/09/04 08:00 PM GMT+2	stored on window	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
check for bitflip	HDD 80GB	H80N1	2018/09/13 06:30 PM GMT+2	no bitflip found						
initial zero writing	USB Card violet	U01N1	2018/09/13 08:30 PM GMT+2	no errors during process						
initial zero writing	USB Card white	U04N2	2018/09/13 08:45 PM GMT+2	no errors during process						
initial zero writing	USB Stick silver	U01N3	2019/09/13 09:00 PM GMT+2	no errors during process						
initial zero writing	SD Card black	C01N4	2020/09/13 09:00 PM GMT+2	no errors during process						
location change	USB Card violet	U01N1	2018/09/13 09:15 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
location change	USB Card white	U04N2	2018/09/13 09:15 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
location change	USB Stick silver	U01N3	2018/09/13 09:15 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
location change	SD Card black	C01N4	2018/09/13 09:15 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
location change	HDD 80GB	H80N1	2018/09/16 07:30 AM GMT+2	traveling VIE-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	7925m	Flight OS111 Vienna,AT to Munich,DE. Flight time 1:05h, Distance 222m
location change	USB Card violet	U01N1	2018/09/16 07:30 AM GMT+2	traveling VIE-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	7925m	Flight OS111 Vienna,AT to Munich,DE. Flight time 1:05h, Distance 222m
location change	USB Card white	U04N2	2018/09/16 07:30 AM GMT+2	traveling VIE-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	7925m	Flight OS111 Vienna,AT to Munich,DE. Flight time 1:05h, Distance 222m
location change	USB Stick silver	U01N3	2018/09/16 07:30 AM GMT+2	traveling VIE-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	7925m	Flight OS111 Vienna,AT to Munich,DE. Flight time 1:05h, Distance 222m
location change	SD Card black	C01N4	2018/09/16 07:30 AM GMT+2	traveling VIE-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	7925m	Flight OS111 Vienna,AT to Munich,DE. Flight time 1:05h, Distance 222m
location change	HDD 80GB	H80N1	2018/09/16 03:30 PM GMT-7	traveling MUC-LAX by plane	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m	11890m	Flight UA8860 Munich,DE to Los Angeles,CA,USA. Flight time: 12:05h, Distance 5978m
location change	USB Card violet	U01N1	2018/09/16 03:30 PM GMT-7	traveling MUC-LAX by plane	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m	11890m	Flight UA8860 Munich,DE to Los Angeles,CA,USA. Flight time: 12:05h, Distance 5978m
location change	USB Card white	U04N2	2018/09/16 03:30 PM GMT-7	traveling MUC-LAX by plane	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m	11890m	Flight UA8860 Munich,DE to Los Angeles,CA,USA. Flight time: 12:05h, Distance 5978m
location change	USB Stick silver	U01N3	2018/09/16 03:30 PM GMT-7	traveling MUC-LAX by plane	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m	11890m	Flight UA8860 Munich,DE to Los Angeles,CA,USA. Flight time: 12:05h, Distance 5978m
location change	SD Card black	C01N4	2018/09/16 03:30 PM GMT-7	traveling MUC-LAX by plane	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m	11890m	Flight UA8860 Munich,DE to Los Angeles,CA,USA. Flight time: 12:05h, Distance 5978m
location change	HDD 80GB	H80N1	2018/09/16 03:30 PM GMT-7	traveling LAX-SAN by train	San Diego,CA,USA	32° 48' 55"N	117° 8' 6"W	20m		6th floor of 45 floor building, surrounded by unknown materials
location change	USB Card violet	U01N1	2018/09/16 03:30 PM GMT-7	traveling LAX-SAN by train	San Diego,CA,USA	32° 48' 55"N	117° 8' 6"W	20m		6th floor of 45 floor building, surrounded by unknown materials
location change	USB Card white	U04N2	2018/09/16 03:30 PM GMT-7	traveling LAX-SAN by train	San Diego,CA,USA	32° 48' 55"N	117° 8' 6"W	20m		6th floor of 45 floor building, surrounded by unknown materials
location change	USB Stick silver	U01N3	2018/09/16 03:30 PM GMT-7	traveling LAX-SAN by train	San Diego,CA,USA	32° 48' 55"N	117° 8' 6"W	20m		6th floor of 45 floor building, surrounded by unknown materials
location change	SD Card black	C01N4	2018/09/16 03:30 PM GMT-7	traveling LAX-SAN by train	San Diego,CA,USA	32° 48' 55"N	117° 8' 6"W	20m		6th floor of 45 floor building, surrounded by unknown materials
location change	HDD 80GB	H80N1	2018/09/27 01:30 PM GMT-7	traveling SAN-LAX by train	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m		
location change	USB Card violet	U01N1	2018/09/27 01:30 PM GMT-7	traveling SAN-LAX by train	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m		
location change	USB Card white	U04N2	2018/09/27 01:30 PM GMT-7	traveling SAN-LAX by train	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m		
location change	USB Stick silver	U01N3	2018/09/27 01:30 PM GMT-7	traveling SAN-LAX by train	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m		
location change	SD Card black	C01N4	2018/09/27 01:30 PM GMT-7	traveling SAN-LAX by train	Los Angeles,CA,USA	33° 56' 33"N	118° 24' 29"W	38m		
location change	HDD 80GB	H80N1	2018/09/28 01:15 PM GMT+2	traveling LAX-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	11890m	Flight LH453 Los Angeles,CA,USA to Munich,DE. Flight time: 12:05h, Distance 5978m
location change	USB Card violet	U01N1	2018/09/28 01:15 PM GMT+2	traveling LAX-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	11890m	Flight LH453 Los Angeles,CA,USA to Munich,DE. Flight time: 12:05h, Distance 5978m
location change	USB Card white	U04N2	2018/09/28 01:15 PM GMT+2	traveling LAX-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	11890m	Flight LH453 Los Angeles,CA,USA to Munich,DE. Flight time: 12:05h, Distance 5978m
location change	USB Stick silver	U01N3	2018/09/28 01:15 PM GMT+2	traveling LAX-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	11890m	Flight LH453 Los Angeles,CA,USA to Munich,DE. Flight time: 12:05h, Distance 5978m
location change	SD Card black	C01N4	2018/09/28 01:15 PM GMT+2	traveling LAX-MUC by plane	Munich,DE	48° 21' 14"N	11° 47' 9.9"E	453m	11890m	Flight LH453 Los Angeles,CA,USA to Munich,DE. Flight time: 12:05h, Distance 5978m
location change	HDD 80GB	H80N1	2018/09/28 04:15 PM GMT+2	traveling MUC-VIE by plane	Vienna,AT	48° 6' 37"N	16° 34' 11"E	183m	7925m	Flight LH2330 Munich,DE to Vienna,AT. Flight time: 1:00h, Distance 222m
location change	USB Card violet	U01N1	2018/09/28 04:15 PM GMT+2	traveling MUC-VIE by plane	Vienna,AT	48° 6' 37"N	16° 34' 11"E	183m	7925m	Flight LH2330 Munich,DE to Vienna,AT. Flight time: 1:00h, Distance 222m
location change	USB Card white	U04N2	2018/09/28 04:15 PM GMT+2	traveling MUC-VIE by plane	Vienna,AT	48° 6' 37"N	16° 34' 11"E	183m	7925m	Flight LH2330 Munich,DE to Vienna,AT. Flight time: 1:00h, Distance 222m
location change	USB Stick silver	U01N3	2018/09/28 04:15 PM GMT+2	traveling MUC-VIE by plane	Vienna,AT	48° 6' 37"N	16° 34' 11"E	183m	7925m	Flight LH2330 Munich,DE to Vienna,AT. Flight time: 1:00h, Distance 222m
location change	SD Card black	C01N4	2018/09/28 04:15 PM GMT+2	traveling MUC-VIE by plane	Vienna,AT	48° 6' 37"N	16° 34' 11"E	183m	7925m	Flight LH2330 Munich,DE to Vienna,AT. Flight time: 1:00h, Distance 222m
location change	HDD 80GB	H80N1	2018/09/28 06:30 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
location change	USB Card violet	U01N1	2018/09/28 06:30 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
location change	USB Card white	U04N2	2018/09/28 06:30 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
location change	USB Stick silver	U01N3	2018/09/28 06:30 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
location change	SD Card black	C01N4	2018/09/28 06:30 PM GMT+2	stored on office table	Leoben, AT	47° 22' 48"N	15° 5' 24"E	554m		2ND Floor of 5 Floor building, surrounded by 36cm perforated brick, concrete ceiling in each floor
check for bitflip	HDD 80GB	H80N1	2018/09/30 06:30 AM GMT+2	no results, logfile: bitflip_H80N1_20180930.log						
check for bitflip	USB Card violet	U01N1	2018/09/30 06:30 AM GMT+2	0.276 GB non zero values stored in logfile: bitflip_U01N1_20180930.log						
check for bitflip	USB Card white	U04N2	2018/09/30 06:30 AM GMT+2	2.4GB non zero values stored in logfile: bitflip_U04N2_20180930.log						
check for bitflip	USB Stick silver	U01N3	2018/09/30 06:45 AM GMT+2	no results, logfile: bitflip_U01N3_20180930.log						
check for bitflip	SD Card black	C01N4	2018/09/30 06:45 AM GMT+2	no results, logfile: bitflip_C01N4_20180930.log						

*ALL TIMESTAMPS REPRESENTING THE END OF THE PROCESS

Figure 3.13: Experiment for detecting bitflip occurrings from high energetic particles in higher altitudes on general storage devices

Chapter 4

Evaluation

This chapter discusses and compares the results of from Braunecker, 2012 and other RNGs with the implementation of the ERBG Android application. A detailed explanation for each test, included in the Statistical Test Suite (e.g. Frequency, Cumulative Sums, et cetera), is provided within Chapter 2.8.

4.1 Test Setup and Environment

In this chapter the environmental setup with all relevant parameters are described. Even though it will *not* be possible to reproduce the exact same results. This is due to factors which, at this moment of time, cannot be manipulated by humans. Some of those factors include, but are not limited to, for instance the shattering of sunlight in the lens of the camera and/or artifacts captured from the image sensor built in the smartphone. Nevertheless, it is essential to describe the used setup in detail to allow a rapprochement.

4.1.1 Devices used in the Experiment

For the execution and gathering of extractable raw random material, two devices were chosen. On the one hand there is the newer, in 2017 released device, Mi A1 from Xiaomi, 2017. On the other hand, there is the in 2015 released Nexus 5X from LG Electronics, Inc., 2015.

Both of them are running stock[1] or near stock [2] Android. Whereas the primary camera for the LG Nexus 5X has a Sony image sensor model IMX377, the Xiaomi Mi A1 has the OV12A10 from OmniVision. When it comes to the secondary camera the LG Nexus 5X has the OmniVision OV5693 as an image sensor, on the other hand the Xiaomi Mi A1 has the S5K5E8 from Samsung built in as a secondary camera image sensor (DeviceSpecifications.com, 2019).

Type/Detail	Xiaomi Mi A1	LG Nexus 5X
Primary Camera / Back Camera		
Image Sensor Type	PureCel	CMOS BSI
Image Sensor Manufacturer	OmniVision	Sony
Image Sensor Model	OV12A10	IMX377
Secondary Camera / Front Camera		
Image Sensor Type	CMOS BSI	CMOS BSI 2
Image Sensor Manufacturer	Samsung	OmniVision
Image Sensor Model	S5K5E8	OV5693

Table 4.1: Comparison of the specifications from the different used devices, LG Nexus 5X as well as Xiaomi Mi A1 (DeviceSpecifications.com, 2019)

4.1.2 Used physical templates

As mentioned in the beginning it may be very difficult to receive the same test results without having the raw material video at hand. During the early stages of development several tests had taken place with one device only. After the usage of another test device, the “LG Nexus 5X - Primary - Stefan” test results of the Statistical Test Suite of NIST by Rukhin et al., 2010 had to be considered mostly as non random. For this reason, the desire to have a comparable test setup arose. Thus, some kind of apparatus that is capable of capturing the same scene at the same time within the same angle and height had to be constructed. This apparatus was found in the self-made carton template as shown in Figure 4.3 and Figure 4.4.

As can be seen in Figure 4.3, there are two holes in the front. For the second hole the device Nexus 5X is already in place and the ERBG Android app is started. It can also be seen that the slit in the front of the carton template allows to start the

[1]The term ’Stock Android’ describes that the operating system is not modified but delivered and installed directly from the publisher Google Inc.
[2]The term ’near stock Android’ describes the program *Android One* powered by Google Inc.

Figure 4.1: The Xiaomi Mi A1 smartphone on which the ERBG app was tested on. Notice that the extraction of random numbers out of the gathered file could also be done on other devices (Xiaomi, 2017)

Figure 4.2: The LG Nexus 5X smartphone on which the ERBG app was tested on. Notice that the extraction of random numbers out of the gathered file could also be done on other devices (LG Electronics, Inc., 2015)

capturing mode. If two devices are placed into this template, it allows the user to simultaneously start both devices by using two fingers, for instance the index and pinky finger. However, the backside of the carton template (see Figure 4.4) has also two holes, but those holes have a very different purpose. In order to capture a video the camera lenses must have a free view, for this reason the holes in the back were made.

When the carton template is in use, the flap is folded down. Then the capturing mode has to be activated by triggering the *record* buttons simultaneously with two fingers. At this point, the carton template is held with both hands so that the flap is held down and the template itself can not fall ot of hands.

4.2 Results

In this section the results for the proposed ERBG algorithm will be discussed and compared to other works. Some approaches and possible use cases like shaking the device by hand or just walk straight ahead are described this section. The direct comparison of the *Fast Forest Circle Spin Test*, which included a humongous amount of motion blur, and the *Meadow Slow Translation Test*, without much motion blur, shows that the motion blur could be the reason for the good (but technically failed) test results of the

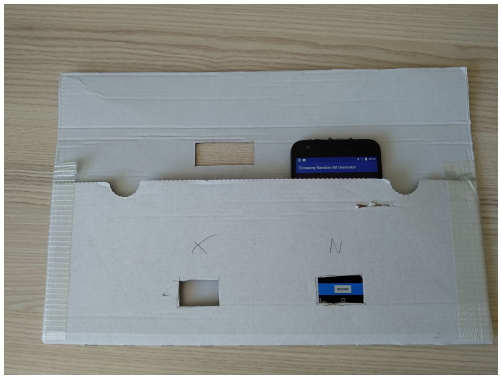

Figure 4.3: Showing the front side of the improvised multi smartphone holder to capture videos with the same probabilities

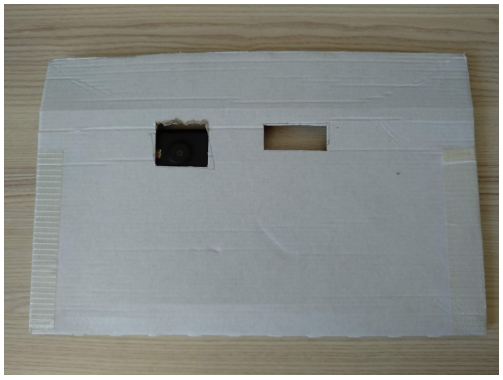

Figure 4.4: Showing the back side of the improvised multi smartphone holder to capture videos with the same probabilities

Statistical Test Suite.

It is worth noting that the NIST Statistical Testsuite marks failed tests with an asterisk '*', this asterisk can also be found within the prepared test result tables. However, some tests have an overall P-value that is below 0.01 or above 0.99 but were not marked as failed by the test suite. Moreover, every subtest of a specific test was within the acceptable range even if the overall P-value was out of that range but not marked as failed. Therefore within this Master Thesis such results are also considered to be passed.

4.2.1 Extracted Randomness from Audio

For capturing the amount of 10^8 random bits from the microphone input, a time of 3:32 hours was needed. However, one test did not pass the Statistical Test Suite. This method of randomness extraction has worked on every tested device within the same quality. Excluding the downside of a long duration and a single failed test this method is very hopeful.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
11	8	12	13	11	9	11	10	5	10	0.867692	100/100	Frequency	YES
11	9	8	17	14	10	10	10	5	6	0.262249	98/100	BlockFrequency	YES
13	11	10	14	7	13	6	10	10	6	0.574903	100/100	CumulativeSums	YES
10	11	9	12	12	14	2	7	12	11	0.319084	100/100	CumulativeSums	YES
10	12	14	11	6	7	8	7	9	16	0.383827	100/100	Runs	YES
7	6	14	13	12	5	8	12	17	6	0.085587	99/100	LongestRun	YES
8	8	9	9	11	8	10	11	15	11	0.897763	99/100	Rank	YES
10	11	7	9	9	7	17	15	5	10	0.213309	98/100	FFT	YES
9	15	7	8	9	13	3	10	6	20	0.010988	100/100	NonOverlappingTemplate	YES
9	11	12	9	10	7	10	9	13	10	0.978072	99/100	OverlappingTemplate	YES
12	11	11	10	8	8	9	10	12	9	0.991468	99/100	Universal	YES
13	11	7	13	9	7	14	8	5	13	0.419021	95/100*	ApproximateEntropy	NO
8	7	4	5	6	7	8	6	5	4	0.949602	57/60	RandomExcursions	YES
5	4	5	8	3	4	10	2	14	5	0.017912	60/60	RandomExcursionsVariant	YES
6	11	5	14	8	11	15	11	11	8	0.401199	100/100	Serial	YES
9	4	10	11	12	7	13	12	12	10	0.657933	99/100	Serial	YES
8	10	11	12	10	11	8	12	8	10	0.987896	100/100	LinearComplexity	YES

Table 4.2: Test results for extracted randomness out of a microphone input taken by the device “Xiaomi Mi A1 - Stefan”

4.2.2 Primary versus Secondary Camera

This section shows the differences that occurred between using the primary camera compared with the secondary camera of the devices used. For this test the secondary camera of the device “Xiaomi Mi A1 - Secondary - Stefan” was used. The idea behind this run is that a better conclusion on whether the image sensor or any other interference causes the Statistical Test Suite to fail on extracted randomness gathered from raw material videos taken by other devices. This can be achieved with a run of devices having the same image sensor type but one device produces randomness and the other device does not.

However, as shown in Table 4.3, all tests were passed and therefore can be considered to be random.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
17	10	8	9	14	7	9	12	6	8	0.319084	97/100	Frequency	YES
16	14	9	9	9	12	7	8	7	9	0.514124	96/100	BlockFrequency	YES
16	9	8	10	12	5	11	16	6	7	0.153763	96/100	CumulativeSums	YES
16	10	10	9	14	10	10	10	6	5	0.401199	97/100	CumulativeSums	YES
12	5	7	8	10	19	14	10	11	4	0.040108	99/100	Runs	YES
15	10	12	6	13	12	7	5	5	15	0.115387	99/100	LongestRun	YES
11	12	15	9	12	10	5	12	6	8	0.494392	100/100	Rank	YES
9	10	6	7	12	15	11	10	14	6	0.455937	100/100	FFT	YES
10	4	9	17	14	10	6	9	5	16	0.035174	99/100	NonOverlappingTemplate	YES
15	8	5	15	17	5	12	7	7	9	0.040108	99/100	OverlappingTemplate	YES
10	12	6	12	13	10	10	14	10	3	0.366918	99/100	Universal	YES
9	8	8	15	10	17	5	6	11	11	0.181557	99/100	ApproximateEntropy	YES
7	11	3	9	5	1	6	9	4	4	0.025193	59/59	RandomExcursions	YES
10	5	5	5	8	4	3	3	11	5	0.071177	58/59	RandomExcursionsVariant	YES
13	9	8	10	16	8	10	13	7	6	0.455937	100/100	Serial	YES
11	11	8	17	9	11	7	8	7	11	0.534146	98/100	Serial	YES
16	6	12	11	8	2	11	11	10	13	0.137282	99/100	LinearComplexity	YES

Table 4.3: Test results for extracted randomness out of a video taken by the device “Xiaomi Mi A1 - Secondary - Stefan”, internally referred as “Forest Fast Circle Spin”

The Table 4.3 shows the results of the tests from the Statistical Test Suite for the extracted randomness from the gathered raw material video that was captured with the secondary camera of the device “Xiaomi Mi A1 - Secondary - Stefan”.

4.2.3 ERBG - Forest Fast Circle Spin

The raw material file was gathered within a forest. During the 2:50 minute long video the capturing device, Xiaomi Mi A1 - Primary - Marie, was often revolved wildly and quickly around the axis of the person filming. This actions gave the final video in some sequences a high amount of motion blur, as can be seen in Figure 4.5. However, not the whole video had this motion blur in it, as Figure 4.6 clearly shows. To conclude this specific test, all 15 tests of the Statistical Test Suite are passed. According to Rukhin et al., 2010, a specific statistical test, with exception of the random excursion and random excursion variant, passes when at least 96 out of 100 bit streams have a P-value > 0.01 but not P-value < 0.99. For the random excursion and random excursion variant the pass rate is at least 59 out of 62 bit streams.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
13	11	13	11	10	11	6	6	7	12	0.678686	100/100	Frequency	YES
12	8	13	12	15	6	9	11	10	4	0.350485	99/100	BlockFrequency	YES
13	15	15	6	7	11	8	10	9	6	0.304126	100/100	CumulativeSums	YES
11	14	11	5	8	12	13	10	9	7	0.637119	100/100	CumulativeSums	YES
11	14	9	14	10	7	9	8	8	10	0.816537	97/100	Runs	YES
8	8	10	7	10	15	11	6	12	13	0.616305	99/100	LongestRun	YES
6	7	11	9	11	19	18	6	10	3	0.004629	100/100	Rank	YES
9	21	11	6	12	6	9	13	7	6	0.021999	100/100	FFT	YES
20	13	10	5	8	8	9	7	9	11	0.080519	97/100	NonOverlappingTemplate	YES
12	7	12	7	14	11	12	9	4	12	0.455937	98/100	OverlappingTemplate	YES
13	14	7	10	6	10	5	11	10	14	0.419021	97/100	Universal	YES
6	9	10	11	9	17	10	9	9	10	0.637119	99/100	ApproximateEntropy	YES
12	1	5	6	7	5	8	7	2	6	0.028817	57/59	RandomExcursions	YES
4	7	4	10	7	3	3	6	5	10	0.12962	58/59	RandomExcursionsVariant	YES
11	12	10	8	10	12	10	4	15	8	0.55442	97/100	Serial	YES
10	8	8	11	14	6	10	14	10	9	0.759756	99/100	Serial	YES
8	11	9	8	9	11	11	9	13	11	0.983453	99/100	LinearComplexity	YES

Table 4.4: Test results for extracted randomness out of a video taken by the device, “Xiaomi Mi A1 - Primary - Marie” internally referred as “Forest Fast Circle Spin”

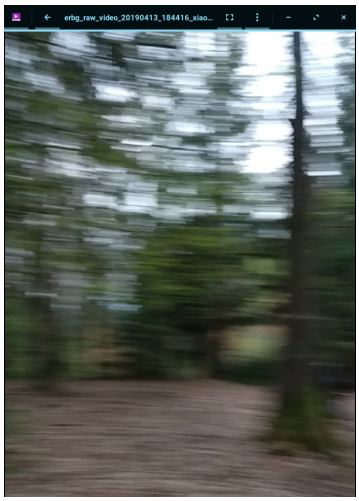

Figure 4.5: A screenshot from the raw material video, of the timestamp 0:11, from which the random numbers were extracted.

Figure 4.6: A screenshot from the raw material video, of the timestamp 1:19, from which the random numbers were extracted.

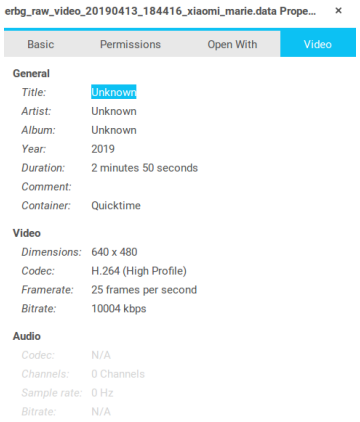

Figure 4.7: A screenshot of the properties of the raw material video from which the random numbers were extracted.

4.2.4 ERBG - Meadow Slow Translation

The raw material was gathered within a meadow on two devices at the exact same time with a physical framework that kept both devices in place. The physical framework

was made from a stiff sheet of carton with holes in the front in order to be able to trigger both devices at the same time. It also had two more holes in the back to provide a free view for the camera lenses. The video was taken with a slow translation, thus a slow forward movement. In contrast to the test results in Chapter 4.2.3, for the "Forest Fast Circle Spin" video, the result for the random numbers out of this video have failed many tests of the Statistical Test Suite, but were very close to success, as shown in Table 4.5.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
34	13	12	7	8	6	8	7	3	2	0*	97/100	Frequency	NO
25	15	5	15	4	10	8	4	6	8	0.000009*	97/100	BlockFrequency	NO
33	13	12	5	9	5	7	5	4	7	0*	96/100	CumulativeSums	NO
34	15	9	6	7	6	6	7	3	7	0*	96/100	CumulativeSums	NO
10	15	11	12	7	8	5	9	10	13	0.55442	100/100	Runs	YES
10	14	9	14	5	10	10	10	8	10	0.719747	99/100	LongestRun	YES
10	5	11	13	17	4	13	8	9	10	0.145326	98/100	Rank	YES
13	8	12	8	10	5	9	8	8	19	0.137282	100/100	FFT	YES
10	8	4	9	13	15	16	8	7	10	0.191687	95/100*	NonOverlappingTemplate	NO
12	10	8	9	11	13	8	12	6	11	0.883171	98/100	OverlappingTemplate	YES
9	8	8	10	10	9	12	11	9	14	0.955835	99/100	Universal	YES
16	7	11	14	7	8	9	13	8	7	0.366918	97/100	ApproximateEntropy	YES
3	5	1	1	4	6	6	3	7	7	0.227773	42/43	RandomExcursions	YES
9	5	7	3	5	3	0	3	5	3	0.113706	42/43	RandomExcursionsVariant	YES
18	8	11	10	9	17	6	6	6	9	0.051942	99/100	Serial	YES
16	7	13	11	12	10	7	6	8	10	0.455937	100/100	Serial	YES
9	17	9	10	15	6	11	6	12	5	0.12962	99/100	LinearComplexity	YES

Table 4.5: Test results for extracted randomness out of a video taken by the device, "Xiaomi Mi A1 - Primary - Stefan" internally referred as "Meadow Slow Translation"

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
99	0	1	0	0	0	0	0	0	0	0*	3/100*	Frequency	NO
100	0	0	0	0	0	0	0	0	0	0*	0/100*	BlockFrequency	NO
100	0	0	0	0	0	0	0	0	0	0*	0/100*	CumulativeSums	NO
100	0	0	0	0	0	0	0	0	0	0*	0/100*	CumulativeSums	NO
99	0	0	0	1	0	0	0	0	0	0*	1/100*	Runs	NO
62	13	3	6	4	4	4	2	2	0	0*	67/100*	LongestRun	NO
12	7	9	9	8	12	18	6	11	8	0.289667	100/100	Rank	YES
12	13	13	9	6	10	11	9	9	8	0.867692	97/100	FFT	YES
99	1	0	0	0	0	0	0	0	0	0*	2/100*	NonOverlappingTemplate	NO
88	3	4	2	0	2	1	0	0	0	0*	20/100*	OverlappingTemplate	NO
60	13	7	6	4	1	4	1	4	0	0*	57/100*	Universal	NO
96	1	0	2	0	0	0	1	0	0	0*	17/100*	ApproximateEntropy	NO
0	0	0	0	0	0	0	0	0	0	—	——	RandomExcursions	NO
0	0	0	0	0	0	0	0	0	0	—	——	RandomExcursionsVariant	NO
69	4	5	6	5	2	4	1	4	0	0*	55/100*	Serial	NO
13	12	6	12	7	13	7	8	12	10	0.657933	96/100	Serial	YES
13	5	9	10	14	8	14	6	6	15	0.171867	100/100	LinearComplexity	YES

Table 4.6: Test results for extracted randomness out of a video taken by the device, "Nexus 5X - Primary - Stefan" internally referred as "Meadow Slow Translation"

Even though both gathered random bit files failed, the device "Xiaomi Mi A1 - Primary - Stefan" missed the tests very close in some cases, as shown in Table 4.5. On the other hand the test results of the device "Nexus 5X - Primary - Stefan" missed the threshold for randomness by far, as can be seen in Table 4.6. When a closer look onto the raw

material is taken, it is visible that the screenshot in Figure 4.9, looks far more clear and calm, it also has a lot more details in it as the screenshot in Figure 4.8.

Figure 4.8: A screenshot from the raw material video taken by “Xiaomi Mi A1 - Primary - Stefan”, of the timestamp 0:49, from which the random numbers were extracted.

Figure 4.9: A screenshot from the raw material video taken by “Nexus 5X - Primary - Stefan”, of the timestamp 0:49, from which the random numbers were extracted.

4.3 Results of other RNGs

To generate a general overview and provide a small basis for comparison, this section will describe results from other RNGs. It was possible to either generate a random sequence with the generator at hand or to have access to the test results. However, all the results were casted/forged to the design used within this Master Thesis at hand.

4.3.1 Chaotic Algorithm Braunecker, 2012

When looking at the results of the two different chaotic algorithms of Braunecker, 2012, as shown in Table 4.7, and Table 4.8, out of all other passed tests the *Non Overlapping Template Test* failed. A test from the Statistical Test Suite by Rukhin et al., 2010, passed when the given threshold of at least 96 successful out of 100 bit streams

was reached. The exception was, that the threshold for random excursion and random excursion variant was at least 59 successful bit streams out of 62 bit streams.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
6	12	8	13	10	12	9	14	6	10	0.637119	99/100	Frequency	YES
13	9	16	10	8	9	8	3	14	10	0.213309	100/100	BlockFrequency	YES
7	12	9	9	9	9	17	11	11	6	0.494392	99/100	CumulativeSums	YES
6	10	12	15	8	10	9	12	10	8	0.759756	99/100	CumulativeSums	YES
13	14	7	15	7	11	11	11	5	6	0.262249	100/100	Runs	YES
8	11	8	13	10	11	10	9	9	11	0.987896	99/100	LongestRun	YES
9	18	6	12	7	5	11	10	12	10	0.191687	100/100	Rank	YES
16	11	15	8	8	6	13	4	7	12	0.108791	99/100	FFT	YES
6	9	4	13	16	12	12	3	17	8	0.013569	98/100	NonOverlappingTemplate	YES
17	13	6	8	10	7	9	12	12	6	0.262249	99/100	OverlappingTemplate	YES
9	10	18	8	5	4	9	11	14	12	0.085587	99/100	Universal	YES
5	14	7	13	8	10	10	11	12	10	0.657933	99/100	ApproximateEntropy	YES
1	8	5	7	7	6	1	11	7	9	0.082177	62/62	RandomExcursions	YES
4	2	7	9	4	9	9	2	9	7	0.162606	62/62	RandomExcursionsVariant	YES
15	10	7	7	13	14	10	6	8	10	0.455937	96/100	Serial	YES
12	11	11	11	15	7	12	5	6	10	0.474986	97/100	Serial	YES
10	8	18	11	13	7	10	8	9	6	0.289667	100/100	LinearComplexity	YES

Table 4.7: Test results from NIST Statistical Test Suite by Rukhin et al., 2010, of extracted random bits from “standard map”, a chaotic algorithm of Braunecker, 2012.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
11	7	10	13	15	12	10	7	9	6	0.595549	100/100	Frequency	YES
12	17	7	9	9	7	10	10	7	12	0.474986	100/100	BlockFrequency	YES
10	9	6	14	20	7	8	14	8	4	0.016717	99/100	CumulativeSums	YES
13	4	12	6	17	11	15	7	10	5	0.042808	100/100	CumulativeSums	YES
7	9	8	12	6	13	12	17	7	9	0.304126	99/100	Runs	YES
13	7	11	14	5	8	10	12	9	11	0.637119	100/100	LongestRun	YES
14	10	9	13	13	4	11	10	5	11	0.366918	98/100	Rank	YES
14	8	8	14	14	8	11	8	7	8	0.55442	99/100	FFT	YES
13	9	7	10	14	7	11	11	8	10	0.834308	94/100*	NonOverlappingTemplate	NO
8	12	8	5	15	7	17	8	6	14	0.075719	100/100	OverlappingTemplate	YES
8	7	9	15	12	7	9	9	13	11	0.699313	100/100	Universal	YES
12	9	9	12	9	6	11	14	8	10	0.851383	97/100	ApproximateEntropy	YES
4	2	6	9	9	6	5	5	13	8	0.078086	63/67	RandomExcursions	YES
1	7	9	3	4	6	13	11	7	6	0.015065	67/67	RandomExcursionsVariant	YES
8	6	12	11	10	8	5	11	12	17	0.289667	99/100	Serial	YES
10	6	11	9	11	8	12	11	13	9	0.924076	99/100	Serial	YES
2	9	11	14	12	14	16	4	4	14	0.00716	100/100	LinearComplexity	YES

Table 4.8: Test results from NIST Statistical Test Suite by Rukhin et al., 2010, of extracted random bits from “logistics map”, a chaotic algorithm of Braunecker, 2012.

4.3.2 QRNG Quantis ID Quantique

It was possible to use the QRNG from ID Quantique for generating random numbers. However, when generating integers the datarate of the QRNG significantly dropped. As it was possible to reach the customer support the conclusion was drawn that the proposed datarate of 4Mb/s only counts for binary extraction and not for integers. Therefore, for this test run a binary sequence was extracted from the QRNG and tested. For testing the binary sequence the configuration of the NIST Statistical Test Suite for the parameter 'Input File Format' was set to 'Binary'; note that all other tests were set to 'ASCII'. The test results for the binary output can be found within Table 4.9.

As the ERBG with its Java Standalone Application was capable of the extraction of randomness from any given file, it was tested on the binary output of the Quantis QRNG too, as can be seen in Table 4.10. An interesting fact is, that two "Non Overlapping Template Tests" failed by testing the genuine binary output of the Quantis QRNG, whereas when processing the same output through the ERBG Java Standalone Application all test passed.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
7	5	14	10	14	13	11	11	7	8	0.437274	100/100	Frequency	YES
12	16	5	11	10	4	13	8	8	13	0.171867	100/100	BlockFrequency	YES
10	4	8	12	9	14	16	9	9	9	0.350485	100/100	CumulativeSums	YES
8	11	8	12	10	14	8	10	10	9	0.946308	100/100	CumulativeSums	YES
14	13	9	9	11	7	6	15	9	7	0.455937	98/100	Runs	YES
15	10	7	12	7	9	12	13	10	5	0.474986	100/100	LongestRun	YES
10	7	12	12	11	10	7	13	10	8	0.911413	100/100	Rank	YES
13	7	10	10	11	11	8	11	8	11	0.964295	97/100	FFT	YES
15	9	7	10	10	13	7	13	9	7	0.616305	95/100*	NonOverlappingTemplate	NO
12	9	15	4	12	17	6	6	11	8	0.075719	95/100*	NonOverlappingTemplate	NO
14	10	6	10	7	7	11	16	6	13	0.262249	99/100	OverlappingTemplate	YES
8	10	13	17	10	12	8	8	7	7	0.419021	100/100	Universal	YES
14	10	13	4	11	12	6	10	6	14	0.249284	99/100	ApproximateEntropy	YES
3	1	9	8	7	3	10	4	5	5	0.071177	55/55	RandomExcursions	YES
2	4	8	4	8	3	11	9	3	3	0.028817	55/55	RandomExcursionsVariant	YES
5	7	10	10	12	8	13	11	12	12	0.739918	100/100	Serial	YES
3	6	7	9	16	10	9	18	13	9	0.028817	100/100	Serial	YES
8	6	10	9	12	16	7	10	10	12	0.595549	97/100	LinearComplexity	YES

Table 4.9: Test results from NIST Statistical Test Suite by Rukhin et al., 2010, of the extracted binary random bits from the Quantis QRNG

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST	PASS
7	5	15	11	11	12	16	4	10	9	0.12962	99/100	Frequency	YES
8	8	7	9	10	8	9	16	9	16	0.383827	100/100	BlockFrequency	YES
6	8	12	13	11	12	7	13	10	8	0.739918	99/100	CumulativeSums	YES
7	11	9	10	11	16	6	7	12	11	0.55442	99/100	CumulativeSums	YES
9	8	15	10	16	9	6	5	10	12	0.262249	99/100	Runs	YES
11	13	9	12	13	7	10	9	4	12	0.595549	99/100	LongestRun	YES
15	11	11	6	8	8	12	13	7	9	0.595549	100/100	Rank	YES
9	6	13	15	11	4	11	12	10	9	0.401199	99/100	FFT	YES
8	11	9	10	9	10	10	13	9	11	0.99425	100/100	NonOverlappingTemplate	YES
10	17	7	8	6	9	8	8	21	6	0.007694	100/100	NonOverlappingTemplate	YES
12	9	10	13	11	7	7	14	11	6	0.678686	99/100	OverlappingTemplate	YES
10	9	6	10	15	7	14	8	11	10	0.616305	100/100	Universal	YES
9	8	13	13	8	8	9	11	15	6	0.595549	100/100	ApproximateEntropy	YES
4	2	7	8	5	0	10	5	5	10	0.020548	55/56	RandomExcursions	YES
4	5	6	6	5	15	2	6	6	1	0.002043	56/56	RandomExcursionsVariant	YES
8	13	10	10	7	11	8	7	7	19	0.181557	99/100	Serial	YES
10	10	9	17	6	6	5	14	8	15	0.085587	100/100	Serial	YES
9	8	15	8	9	10	10	8	10	13	0.851383	100/100	LinearComplexity	YES

Table 4.10: Test results from the NIST Statistical Test Suite from Rukhin et al., 2010, of the extracted binary and through ERBG processed random bits from the Quantis QRNG

Chapter 5

Conclusion and Outlook

The idea behind randomness has been in existence since the Middle Ages were 'randomness' was tried to be achieved with astragali, i.e. small bones with signs on it, because it was believed that they could tell the future. The meaning behind the term 'randomness' has changed over time. Today the term 'randomness' is defined as the lack of patterns and predictability. However, it is very challenging to create a non-deterministic random bit generator by just throwing some bones like in the Middle Ages.

As by today's definition there are many different aspects to be considered when creating a new Random Bit Generator. This Master Thesis proofs that even with a simple approach it is possible to generate strong randomness out of smartphone sensors like the camera or microphone. By using such smartphone sensors the input of a Random Bit Generator cannot be considered as non-deterministic because as shown in Chapter 3.2.2, the Economy Random Bit Generator (ERBG) Java Standalone Application is extracting randomness out of the gathered raw material video and produces the exact same output by being fed with the same input, which is deterministic by definition.

However, it has been shown in Chapter 4.2, that it is possible to generate a sequence of random bits which can withstand the NIST Statistical Test Suite by Rukhin et al., 2010. In contrast to that, a severe yet unsolved issue occurred as described in Chapter 3.5.4, when the raw material videos are gathered from other devices. This issue could not have been solved as it is not linked to any video configuration like framerate, codec, codec profile or resolution nor the provided stabilization by the operating system nor the image sensor type. In contrast to that, the extraction of randomness from audio was successful with all types of devices used within the Master Thesis but also has the downside of one failed test and a long duration of gathering the random numbers.

When the results of the ERBG are compared with other RBGs it turned out that the binary output of the Quantis QRNG does not always succeed all statistical tests. Moreover, all statistical test have succeeded after the same binary output of the QRNG has been processed through the proposed algorithm within ERBG.

5.1 Outlook

As it was possible to prove that randomness can be extracted from smartphone senors the topic needs even further and much deeper investigation. Moreover, the experiment on cosmic rays had strong results but was not trustworthy due to some errors in the execution. Cosmic rays as entropy source are not predictable due to the fact that they hit our earth's atmosphere with the speed of light and decay within their mean lifetime or half-life. In order to predict incoming cosmic rays a constant of the universe would have to be broken by going beyond the speed of light and also determine when exactly a particle decays within their half-life. However, it is possible to detect single muons with the sensitive CMOS sensor of smartphones as Whiteson et al., 2015, has shown with the smartphone application called "CRAYFIS" in his scientific work.

Therefore, a possible improvement of ERBG could be to implement the "CRAYFIS" muon detection of Whiteson et al., 2015, and extend the given true randomness by interweaving it with the random extraction process on smartphones. With the interweaving of muons it would be possible to achieve an Non-Deterministic Random Bit Generator on smartphones.

Appendix A

Source Code Snippets

A.1 Economy Random Bit Genereator - Basic Statistic Output (EBSO)

The **E**conomy Random Bit Generator - **B**asic **S**tatistic **O**utput gives basic information on the distribution of zeros and ones within a given file of extracted randomness, as also described in Chapter 3.2.1. It is written as a bash script and linked as a symbolic link within */usr/bin*. Unfortunately, as of now, it does not calculate the longest run of either zeros or ones like the final result output from the Android or Java Standalone ERBG does.

```
1 ignis@Sequoia:~\$ ls -la /usr/bin/ebso
2 lrwxrwxrwx 1 root root 39 Mar 16 12:55 /usr/bin/ebso ->
3 /home/ignis/install/erbg\_basic\_stats.sh
```

Listing A.1: Terminal output of the linked location for the helper tool Economy Random Bit Generator - Basic Statistic Output (EBSO)

Figure A.1: Location of the Economy Random Bit Generator - Basic Statistic Output helper script

```
1 #!/bin/sh
```

```
2 if [ $# -eq 0 ]
3   then
4     echo "Economy Random Bit Generator - Basic Statistic Output (EBSO)"
5     echo "\tusage: ebso <filename>\n"
6     echo "Computes basic information about a given file containing one
7     and zero characters. This output is written to console as well as
8     into a file called erbg_basic_stats.txt \n"
9     echo "(c) Stefan Kutschera, BSc [dmz.stefan.kutschera@gmail.com]"
10    return
11 fi
12
13 log(){
14         echo "$1" | tee -a $logfile
15 }
16
17 #defines outputpath for logfile and its name
18 logfile="$PWD/erbg_basic_stats.txt"
19
20 # do some math
21 cntZero=$(grep -o -i 0 $1 | wc -l)
22 cntOne=$(grep -o -i 1 $1 | wc -l)
23 overallSize=$(echo "${cntZero}+${cntOne}" | bc -l)
24
25 ratioZeroOne=$(echo "scale=12; ${cntZero}/${cntOne}" | bc)
26
27 diff=$(echo "scale=12; ${cntZero}-${cntOne}" | bc)
28
29
30 diff_percentage=$(echo "scale=12; (${diff}/${overallSize})*100" | bc)
31
32 # write the results to stdout and the defined logfile
33 log "Analysis and prompting basic statistics for file:"
34 log "$PWD/$1"
35 log "$(date)"
36 log "------------------------------------------------"
37 log "              Distribution of 0/1"
38 log "------------------------------------------------"
39 log "# Overall: \t\t\t $overallSize"
40 log "# '0' : \t\t\t $cntZero"
41 log "# '1' : \t\t\t $cntOne"
42 log "# Difference \t\t\t  $diff"
43 log "% Diff to Overall \t\t $diff_percentage"
44 log "------------------------------------------------"
45 log "Distribution of 0/1 : \t\t $ratioZeroOne"
46 log "------------------------------------------------"
```

```
47 log "###############################################################"
48 log "################################################################\n"
```

A.2 Cleanup Script

In order to clean eventually existing results from previous test runs of the NIST Statistical Test Suite the script as shown in Listing A.2 was executed by the used command shown in Listing A.2.

```
1 ignis@Sequoia:~\$ cd experiments/ && ./cleanup\_results.sh
2 && cd .. && ./assess 1000000
```

Listing A.2: Terminal command for cleaning from may existing previous results

```
1  #!/bin/bash
2
3  # Delete old Directory Structure
4  for dname in AlgorithmTesting BBS CCG G-SHA1 LCG MODEXP MS QCG1 QCG2 XOR ; do
5          rm -r $dname/Frequency
6          rm -r $dname/BlockFrequency
7          rm -r $dname/Runs
8          rm -r $dname/LongestRun
9          rm -r $dname/Rank
10         rm -r $dname/FFT
11         rm -r $dname/NonOverlappingTemplate
12         rm -r $dname/OverlappingTemplate
13         rm -r $dname/Universal
14         rm -r $dname/LinearComplexity
15         rm -r $dname/Serial
16         rm -r $dname/ApproximateEntropy
17         rm -r $dname/CumulativeSums
18         rm -r $dname/RandomExcursions
19         rm -r $dname/RandomExcursionsVariant
20 done
21
22 rm -r AlgorithmTesting/finalAnalysisReport.txt
23 rm -r AlgorithmTesting/freq.txt
24
25 # Recreate Directory Structure
26 for dname in AlgorithmTesting BBS CCG G-SHA1 LCG MODEXP MS QCG1 QCG2 XOR ; do
27         mkdir $dname/Frequency
28         mkdir $dname/BlockFrequency
29         mkdir $dname/Runs
```

```
30          mkdir $dname/LongestRun
31          mkdir $dname/Rank
32          mkdir $dname/FFT
33          mkdir $dname/NonOverlappingTemplate
34          mkdir $dname/OverlappingTemplate
35          mkdir $dname/Universal
36          mkdir $dname/LinearComplexity
37          mkdir $dname/Serial
38          mkdir $dname/ApproximateEntropy
39          mkdir $dname/CumulativeSums
40          mkdir $dname/RandomExcursions
41          mkdir $dname/RandomExcursionsVariant
42 done
```

A.3 Economy Random Bit Generator - Java Standalone Application

This section shows the source code of the later on extracted algorithm of **E**conomy **R**andom **B**it **G**enerator (ERBG) in form of a Java Standalone Application. It was necessary to fulfill this step as the prototype of the ERBG had some issues on Android as it sometimes declined to extract the random numbers out of a previous generated video file.

```
1  package at.fhj.masterthesis.erbg;
2
3  import java.io.*;
4  import java.text.SimpleDateFormat;
5  import java.util.Calendar;
6
7  public class Main {
8
9    private String filename;
10   private String inputFilePath;
11   private String outputFilePath;
12   private String sessionId;
13   private int zero=0;
14   private int one=0;
15   private int firstBytesDumpCounter=0;
16   private int actRunZero=0;
17   private int actRunOne=0;
18   private int longestRunZero=0;
19   private int longestRunOne=0;
```

```
20  public static final String LOG_TAG = "ERBG_JAVA";
21  public static final int FIRST_BYTE_DUMP_THRESHOLD = 4000;
22  Log log=new Log(Log.INFO);
23
24  public Main(String inputFilePath, String outputFilePath)  {
25    setInputFilePath(inputFilePath);
26    setOutputFilePath(outputFilePath);
27    System.out.println(this.outputFilePath);
28    try {
29      generateFromFile();
30    } catch (IOException e) {
31      e.printStackTrace();
32    }
33    checkDistribution();
34  }
35
36  public static void main(String[] args) {
37    if (args.length<1){
38      System.out.println("usage: erbg_standalone <inputfilepath>" +
39          " [outputfilepath]");
40      return;
41    }
42    String inputFilePath=args[0];
43    String outputFilePath=null;
44    if (args.length>1)
45      outputFilePath=args[1];
46    Main main = new Main(inputFilePath,outputFilePath);
47
48  }
49
50  /**
51   *
52   * @param inputFilePath
53   */
54  public void setInputFilePath(String inputFilePath) {
55    this.inputFilePath = inputFilePath;
56  }
57
58  /**
59   *
60   * @param outputFilePath
61   */
62  public void setOutputFilePath(String outputFilePath) {
63    if (outputFilePath==null)
64      this.outputFilePath= System.getProperty("user.dir")+
```

```
65          "/"+getFinalOutputFileName("javaStandalone");
66    else
67      this.outputFilePath = outputFilePath;
68  }
69
70  /**
71   * From the given inputFilePath (i.e. the RAW file) the byte are
72   * reading in and handed over to for random extraction to
73   * 'byteArray2binary(byte[] input)'
74   *
75   * @throws IOException
76   */
77  private void generateFromFile()throws IOException {
78    log.d(LOG_TAG,"Start gathering Data from File");
79    File file = new File(inputFilePath);
80    InputStream ios = null;
81    try {
82      byte[] buffer = new byte[409600];
83      ios = new FileInputStream(file);
84      while (ios.read(buffer) != -1) {
85        this.byteArray2binary(buffer);
86      }
87    }finally {
88      try {
89        if (ios != null)
90          ios.close();
91      } catch (IOException e) {
92        log.e(LOG_TAG,"Ups something with closing went wrong");
93      }
94    }
95    log.d(LOG_TAG,"Finished data gathering from file");
96  }
97
98  /**
99   * This is more or less the heart of the ERBG. It extracts zero or
100  * one by simply calculating 'byte modulo(2)' from any given byte array.
101  * This Method is normally called by the method 'generateFromFile()'
102  *
103  * @param input
104  */
105 private void byteArray2binary(byte[] input){
106   log.d(LOG_TAG,"There are " +input.length +" Bytes to process");
107   StringBuilder sb = new StringBuilder();
108   for (byte b:input) {
109     if (firstBytesDumpCounter > FIRST_BYTE_DUMP_THRESHOLD) {
```

```
110         int i = Math.abs(b)%2;
111         if (i==0) {
112           zero++;
113           actRunZero++;
114           actRunOne = 0;
115           if(actRunZero>longestRunZero) {
116             longestRunZero = actRunZero;
117           }
118         }
119         else {
120           one++;
121           actRunOne++;
122           actRunZero = 0;
123           if(actRunOne>longestRunOne) {
124             longestRunOne = actRunOne;
125           }
126         }
127         sb.append(i);
128       }
129       firstBytesDumpCounter++;
130     }
131     sb.append("\n");
132     writeToFileExternal(sb);
133   }
134
135
136   /**
137    * When all the data from an RAW file is processed this method is called
138    * and writes the given content from the StringBuilder to the given
139    * outputFilePath in append-mode
140    *
141    * @param content
142    */
143   private void writeToFileExternal(StringBuilder content){
144     File file = new File(outputFilePath);
145     try {
146       FileOutputStream f = new FileOutputStream(file,true);
147       f.write(content.toString().getBytes());
148       f.close();
149       log.d(LOG_TAG,"File saved:" + outputFilePath);
150     } catch (FileNotFoundException e) {
151       e.printStackTrace();
152       log.e(LOG_TAG, "******* File not found. Did you" +
153           " add a WRITE_EXTERNAL_STORAGE permission to the manifest?");
154       log.e(LOG_TAG,"Failed to save file" + outputFilePath);
```

```
155     } catch (IOException e) {
156       e.printStackTrace();
157       log.e(LOG_TAG,"Failed to save file" + outputFilePath);
158     }
159   }
160
161   /**
162    * Reads a file from the given filepath and calculates the distribution
163    * of one' and zero's.
164    *
165    */
166   public void checkDistribution() {
167     String[] result = new String[10];
168
169     double ratio = ((double) zero / (double) one) * 1.00000;
170     int difference = (zero-one);
171     int overall = zero+one;
172     System.out.println("-------------------------------------------------");
173     System.out.println("File:" + inputFilePath);
174     System.out.println("Time:"+ sessionId);
175     System.out.println("-------------------------------------------------");
176     System.out.println("# Overall :\t\t"+Integer.toString(overall));
177     System.out.println("# 0 : \t\t\t"+Integer.toString(zero));
178     System.out.println("# 1 : \t\t\t"+Integer.toString(one));
179     System.out.println("# Difference:\t\t"+Integer.toString(difference));
180     System.out.println("% Diff to Overall\t"+Double.toString(((double)
181         difference/overall)*100.00000));
182     System.out.println("-------------------------------------------------");
183     System.out.println("Distribution: \t\t"+Double.toString(ratio));
184     System.out.println("-------------------------------------------------");
185     System.out.println("# Longest Run 0: \t" + longestRunZero);
186     System.out.println("# Longest Run 1: \t" + longestRunOne);
187     System.out.println("#################################################\n\n");
188   }
189
190   /**
191    * Generates or gets the filename for the generated random file that contains
192    * just 0 or 1's.
193    *
194    * @param prefix
195    * @return
196    */
197   public String getFinalOutputFileName(String prefix) {
198     if (!(this.filename == null))
199       return this.filename;
```

```
200     this.filename = "erbg_" + prefix + "_" + getSessionID() + ".data";
201     return this.filename;
202   }
203
204
205   /**
206    * Generates or gets the SessionID which is simply a timestamp.
207    * This is needed for human readable purpose to connect the raw files with
208    * the generates random files.
209    *
210    * @return
211    */
212   private String getSessionID(){
213     if (sessionId==null || sessionId==""){
214       String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").
215           format(Calendar.getInstance().getTime());
216       this.sessionId=timeStamp;
217     }
218     return this.sessionId;
219   }
220
221 }
222
223 class Log{
224
225   public static int ERROR      = 0;
226   public static int INFO       = 1;
227   public static int DEBUG      = 2;
228   public int logLevel;
229
230   public Log(int logLevel) {
231     this.logLevel = logLevel;
232   }
233
234   void i (String tag, String msg){
235     if (logLevel>=Log.INFO)
236       System.out.println("[INFO] "+tag+" : "+msg);
237   }
238
239   void d (String tag, String msg){
240     if (logLevel>=Log.DEBUG)
241       System.out.println("[DEBUG] "+tag+" : "+msg);
242   }
243
244   void e (String tag, String msg){
```

```
245     if (logLevel>=Log.ERROR)
246       System.out.println("[ERROR] "+tag+" : "+msg);
247   }
248
249 }
```

Listing A.3: Source code of the Java Standalone Application of the Economy Random Bit Generator (ERBG)

Appendix B

Changelog

B.1 Changelog

To my fortune, I had an unexpected opportunity to get additional feedback on my thesis from an english native speaker. Unfortunately, it was after the submission-deadline but before the print-deadline. Therefore, I decided to implement the grammar based feedback. The implementation of the feedback was done **without changing** any substantial parts of this thesis. Nevertheless, it was necessary to list which changes were made in full detail. I wrote this thesis by using Latex and git as a code-repository. Hence, it was a simple task to get a very detailed changelog by using the command shown in Listing B.1.

```
1     ignis@Sequoia:~\$ git diff -U0 ./ > documents/diff.txt
```

Listing B.1: Used command for differences between two git commits

```
1 diff --git a/thesis/chapters/01_titlepage.tex b/thesis/chapters/01
    _titlepage.tex
2 index 7a4b9d8..2071ba0 100644
3 --- a/thesis/chapters/01_titlepage.tex
4 +++ b/thesis/chapters/01_titlepage.tex
5 @@ -46 +46 @@
6 -{\bf Master Thesis}\\
7 +{\bf Master's Thesis}\\
8
9
10 diff --git a/thesis/chapters/03_abstract.tex b/thesis/chapters/03
    _abstract.tex
11 index 18943bd..4517582 100644
```

```
12 --- a/thesis/chapters/03_abstract.tex
13 +++ b/thesis/chapters/03_abstract.tex
14 @@ -34 +34 @@
15 -    based muon detection. In order to fulfil the research question
       regarding the creation of a non-deterministic RNG
16 +    based muon detection. In order to fulfill the research question
       regarding the creation of a non-deterministic RNG
17
18
19 diff --git a/thesis/chapters/10_introduction.tex b/thesis/chapters
       /10_introduction.tex
20 index 6149e3d..43869fc 100644
21 --- a/thesis/chapters/10_introduction.tex
22 +++ b/thesis/chapters/10_introduction.tex
23 @@ -12,3 +12,3 @@
24 -In cryptography everything is about secure communication.
       Encryption is when a plaintext and a key are forged
25 -into a chiphertext which can be decrypted by a symmetric- or
       asymmetric key. Moreover, not only information should be
26 -unreadable to anyone without a proper key, excellent cryptography
       is when the algorithm and methodology of the
27 +One of the main focuses of cryptography is about secure
       communication. Encryption is when a plaintext and a key are
       forged
28 +into a chiphertext which can be decrypted by a symmetric- or
       asymmetric key. Information should be
29 +unreadable to anyone without a proper key. Excellent cryptography
       is when the algorithm and methodology of the
30 @@ -19,2 +19,2 @@
31 -not dice. In fact Einstein predicted theories about how the
       universe should work, which were later proven to be correct.
32 -However, in this particular matter it seems, based on what we know
       today, that he was not right at all. For instance,
33 +not play dice. In fact, Einstein predicted theories about how the
       universe should work, which were later proven to be correct.
34 +In this particular matter it seems, based on what we know today,
       that he was not right at all. For instance,
35 @@ -34 +34 @@
36 -also to private individuals. With a price tag of 1.200 USD and
       being quite big in size a QRNG does not fit in our pockets and
37 +also to private individuals. With a price tag of 1.200 USD and
       being quite huge in size a QRNG does not fit in our pockets and
38 @@ -36,2 +36,2 @@
39 -This fact was the inspiration of my Master Thesis to develop an RNG
       . An RNG based on everyday carry-on items like our
```

```
40 -smartphones. Already available sensors on our smartphones could be
       used to get the necessary entropy source, for example
41 +This fact was the inspiration of my Master Thesis to develop a RNG.
        A RNG based on everyday carry-on items like our
42 +smartphones. Already available sensors on our smartphones could be
       used to receive the necessary entropy source, for example
43 @@ -56 +56 @@
44 -Random number generators play a huge role in our everyday life,
       even though we do not recognize them often. Behind a
45 +Random number generators play a huge role in our everyday life,
       even though we do not recognize them. Behind a
46 @@ -59 +59 @@
47 -In order to address this problem, this Master Thesis will try to
       create a prototype of an widely available
48 +In order to address this problem, this Master Thesis will try to
       create a prototype of a widely available
49 @@ -62 +62 @@
50 -providing a prototype of a cheap but strong, thus economical,
       source of an RNG.\par
51 +providing a prototype of a cheap but strong, thus economical,
       source of a RNG.\par
52 @@ -68,2 +68,2 @@
53 -internally. Another process will, with the proposed algorithm in
       this Master Thesis, extract the randomness of the
54 -so-called raw file into a newly created file. This new file
       containing the extracted randomness will consist of only
55 +internally. Another process , with the proposed algorithm in this
       Master Thesis, extract the randomness of the
56 +so-called raw file into a newly created file. This new file
       containing the extracted randomness consist of only
57 @@ -78 +78 @@
58 -applications and networks. Furthermore, both clients will need a
       common set of keys which need to be exchanged once.\vspace{0.5cm}
59 +applications and networks. Furthermore, both clients need a common
       set of keys which need to be exchanged once.\vspace{0.5cm}
60 @@ -95 +95 @@
61 -The Table \ref{tab:msg_prerequirements}, is a proposal of the
       prerequirements of what is needed for Bob\footnote{Alice,
62 +Table \ref{tab:msg_prerequirements}, is a proposal of the
       prerequirements of what is needed for Bob\footnote{Alice,
63 @@ -114 +114 @@
64 -Bob knows that his network is permanently monitored and sends two
       messages to Alice, as be seen in table \ref{tab:msg}.
65 +Bob knows that his network is permanently monitored and sends two
       messages to Alice, as can be seen in Table \ref{tab:msg}.
```

```
66 @@ -125,2 +125,2 @@
67 -source of entropy. The most relevant to this Master Thesis at hand
       will be included. Those existing RNGs within the
68 -literature will be discussed in Chapter \ref{SEC_random_audio_video
       }. One of them examined the quality of randomness
69 +source of entropy. The most relevant one for this Master Thesis are
        included. Those existing RNGs are discussed in
70 +Chapter \ref{SEC_random_audio_video}. One of them examines the
       quality of randomness
71 @@ -129,2 +129,2 @@
72 -$b \equiv \pmod{2}$. Furthermore, most of them operate on a single
       color channel. \par
73 -The Master Thesis at hand uses the knowledge and experience out of
       existing literature and provides an approach that is
74 +$b \equiv \pmod{2}$. In addition, most of them operate on a single
       color channel. \par
75 +The Master Thesis uses the knowledge and experience based on
       existing literature and provides an approach that is
76
77
78 diff --git a/thesis/chapters/20_related.tex b/thesis/chapters/20
       _related.tex
79 index 4e97bc1..05662ba 100644
80 --- a/thesis/chapters/20_related.tex
81 +++ b/thesis/chapters/20_related.tex
82 @@ -7 +7 @@
83 -about common domain-specific terms and standards. Finally, it
       describes what random stands for.
84 +about common domain-specific terms and standards. Finally, it
       describes what random stands for in detail.
85 @@ -20,3 +20,3 @@
86 -From an informal point of view 'random' is used to explain too
       complex causes or causes that are not fully explainable by known
       laws.
87 -From the formal point of view, the term 'random' is the lack of
       predictability or patterns. As an example
88 -for predictability let us assume we collect every tenth result of a
        coin tossing experiment. If the list of coin tossing results is
89 +From an informal point of view 'random' is used to explain too
       complex causes or causes that are not fully explainable by known
       physical laws.
90 +From the formal point of view, the term 'random' is the lack of
       predictability or patterns. For an example of predictability
91 +every tenth result of a coin tossing experiment is collected. If
       the list of coin tossing results is
```

```
92 @@ -29 +29,2 @@
93 -Therefore it can be said that random is a sequence which can not be
       predicted and has a maximum of information in it.
94 +Hence, it can be argued that firstly random is a sequence which can
       not be predicted.
95 +Secondly random has a maximum of information in included, this
      concludes that it cannot be compressed.
96 @@ -54 +55 @@
97 -As we learned what random meant in the Middle Ages and how the term
       'random' is defined by today, we have also learned
98 +As we learned what random meant in the Middle Ages and how the term
       'random' is defined today, we have also seen
99 @@ -57,2 +58,3 @@
100 -By external sources, in perspective of software development, it is
      meant to not use java.random or any other built-in
101 -Pseudo Random Number Generator (PRNG). The problem with those
      Pseudo Random Number Generators is that when they are fed with
      the same
102 +Taking the perspective of software development external sources are
       considered in order to not use java.random or any other built-in
103 +Pseudo Random Number Generator (PRNG).
104 +The problem with those Pseudo Random Number Generators is that when
       they are fed with the same
105 @@ -60 +62 @@
106 -random numbers as before. Therefore built-in Random Number
      Generators are deterministic, the same output can be expected
107 +random numbers as before. Therefore, built-in Random Number
      Generators are deterministic, the same output can be expected
108 @@ -72 +74 @@
109 -signals which travel to the brain and get processed by it, see
      Figure~\ref{PIC_human_ear}.
110 +signals which travel to the brain and is processed by it, see
      Figure~\ref{PIC_human_ear}.
111 @@ -74 +76 @@
112 -and background noise which will be stored in the short-term memory
      .\par
113 +and background noise which is stored in the short-term memory.\par
114 @@ -86 +88 @@
115 -One approach is to work with magnetism, where a small coil is
      placed near a magnet which generates a magnetic field.
116 +one approach is to work with magnetism, where a small coil is
      placed near a magnet which generates a magnetic field.
117 @@ -105 +107 @@
118 -It is now clear what randomness meant in earlier stages of history
      compared to today's definition. The
```

```
119 +Now it is clear what randomness meant in earlier stages of history
      compared to today's definition. The
120 @@ -124,6 +126,6 @@
121 -wavelengths of 400-700 nanometers, is produced, a simply hydrogen
      atom is taken to describe that process.
122 -The hydrogen atom has only one electron within its shell which is
      revolving in one of a limited number of circular
123 -orbits around the nucleus of the hydrogen atom. The electron is in
      its so-called ground state when it is the closest to the
124 -nucleus. On the other hand, there is the state of excitation, in
      which the electron 'jumps' to an orbit further away of
125 -the nucleus of the hydrogen atom. To achieve this state of
      excitation the hydrogen atom has to absorb a certain amount of
      energy,
126 -which can be done by, for example, heating the hydrogen. However,
      as visualized in Figure~\ref{PIC_bohr_model_of_atoms},
127 +wavelengths of 400-700 nanometers, is produced, a simply hydrogen
      atom is taken to describe that process. The hydrogen
128 +atom has only one electron within its shell which revolts in one of
       a limited number of circular orbits around the
129 +nucleus of the hydrogen atom. The electron is in its so-called
      ground state when it is the closest to the nucleus.
130 +On the other hand, there is the state of excitation, in which the
      electron 'jumps' to an orbit further away of
131 +the nucleus of the hydrogen atom. To achieve this state of
      excitation the hydrogen atom has to absorb a certain amount
132 +of energy. For example, this can be done by heating the hydrogen
      atom. However, as visualized in Figure~\ref{
      PIC_bohr_model_of_atoms},
133 @@ -132 +134 @@
134 -of the released photon, thus electromagnetic wave, is in relation
      to the energy transition. The result of this physical process
135 +of the released photon, thus the electromagnetic wave, is in
      relation to the energy transition. The result of this physical
      process
136 @@ -146 +148 @@
137 -    \includegraphics[width=0.4\textwidth]{images/bohrModelOfAtoms.
      png}
138 +    \includegraphics[width=0.3\textwidth]{images/bohrModelOfAtoms.
      png}
139 @@ -155,10 +157,9 @@
140 -like our sun. A source of light also has a specific spectrum type.
      As can be seen in Figure~\ref{PIC_spectral_types}, the
141 -continuous spectrum is a form of genuine light from a source with
      the whole spectrum of wavelengths.
```

```
142 -Moreover, an absorption spectrum is when light travels through a
        thermally cooler
143 -cloud of gas, some photons, with an appropriate level of energy to
        some atoms in that gas, get absorbed. Although
144 -they may be re-emitted, those wavelengths are missing for the
        observer on the other side of the gas cloud, therefore the
145 -absorption spectrum is also called dark line spectrum. On the other
         hand, an emission spectrum is when a gas is
146 -triggered and atoms are brought to a state of so-called excitation.
         In that state of excitation those atoms, specific
147 -for that gas, start to emit light in an related wavelength. Only
        those wavelengths are visible in the spectrum of the
148 -observer which is why it is also called bright line spectrum. All
        three of the discussed spectra's can be seen in
149 -Figure~\ref{PIC_spectral_types}, (\cite[p.172-176]{openstax:2019}),
         (\cite{smith:1999}).
150 +like our sun. A source of light usually has a specific spectrum
        type. Firstly, the continuous spectrum is a form of
151 +genuine light from a source with the whole spectrum of wavelengths,
         as can be seen in Figure~\ref{PIC_spectral_types}.
152 +Secondly, an absorption spectrum is when light travels through a
        thermally cooler cloud of gas and photons with an
153 +appropriate level of energy are absorbed. Although they may be re-
        emitted, those wavelengths are missing for the
154 +observer on the other side of the gas cloud, therefore the
        absorption spectrum is also called dark line spectrum.
155 +Thirdly, an emission spectrum is when a gas is triggered and atoms
        are brought to a state of so-called excitation.
156 +In that state of excitation those atoms, specific for that gas,
        start to emit light in a related wavelength. Only those
157 +wavelengths are visible in the spectrum of the observer which is
        why it is also called bright line spectrum. All three
158 +of the discussed spectra's can be seen in Figure~\ref{
        PIC_spectral_types}, (\cite[p.172-176]{openstax:2019}), (\cite{
        smith:1999}).
159 @@ -174,2 +175,2 @@
160 -To finally see, electromagnetic waves, thus photons emitted by
        electrons switching back from state of excitation to ground state
        ,
161 -have to travel through the cornea and the lens of our eyes, as can
        be seen in Figure~\ref{PIC_cross_section_eye}.
162 +To finally see, electromagnetic waves, thus the photons emitted by
        electrons switching back from the state of excitation
163 +to the ground state, have to travel through the cornea and the lens
         of our eyes, as can be seen in Figure~\ref{PIC_cross_section_eye
```

```
    }.
@@ -632 +633 @@
-In this section the 15 different tests from the test suite provided by the National Institute of Standard and Technology
+In this section, the 15 different tests from the test suite provided by the National Institute of Standard and Technology


diff --git a/thesis/chapters/80_evaluation.tex b/thesis/chapters/80_evaluation.tex
index ca9d35b..4e65852 100644
--- a/thesis/chapters/80_evaluation.tex
+++ b/thesis/chapters/80_evaluation.tex
@@ -5,5 +5,3 @@
-
-In this chapter the different results of the implementation of the ERBG Android application are discussed and
-compared with each other as well as with the results from \cite{braunecker:2012}. A detailed explanation for each
-test, included in the Statistical Test Suite (e.g. Frequency
-, Cumulative Sums, et cetera), is provided within Chapter~\ref{SEC_statistical_tests}.\par
+This chapter discusses and compares the results of from \cite{braunecker:2012} and other RNGs with the implementation
+of the ERBG Android application. A detailed explanation for each test, included in the Statistical Test Suite
+(e.g. Frequency, Cumulative Sums, et cetera), is provided within Chapter~\ref{SEC_statistical_tests}.\par
@@ -13,4 +11,4 @@
-to reproduce the exact same results due to factors which, at this moment of time, cannot be manipulated by humans, it
-is essential to describe the used setup highly detailed to allow a rapprochement. Some of those factors include, but
-are not limited to, for instance the shattering of sunlight in the lens of the camera and/or artifacts captured from
-the image sensor built in the smartphone.
+to reproduce the exact same results. This is due to factors which, at this moment of time, cannot be manipulated by humans.
+Some of those factors include, but are not limited to, for instance the shattering of sunlight in the lens of the
+camera and/or artifacts captured from the image sensor built in the smartphone. Nevertheless, it is essential to describe
+the used setup in detail to allow a rapprochement.
@@ -78 +76 @@
-As mentioned in the beginning it may be very hard to get the same
```

```
    test results without having the raw material video at
193 +As mentioned in the beginning it may be very difficult to receive
    the same test results without having the raw material video at
194 @@ -81 +79 @@
195 -had  to be considered mostly as non random. For this reason, the
    desideratum to have a comparable test setup arised.
196 +had  to be considered mostly as non random. For this reason, the
    desire to have a comparable test setup arose.
197 @@ -84 +82 @@
198 -Figure \ref{fig:carton_template_front}, and Figure~\ref{fig:
    carton_template_back}.
199 +Figure~\ref{fig:carton_template_front} and Figure~\ref{fig:
    carton_template_back}.
200
201
202 diff --git a/thesis/chapters/90_conclusion.tex b/thesis/chapters/90
    _conclusion.tex
203 index 2a0a470..0bbeaa7 100755
204 --- a/thesis/chapters/90_conclusion.tex
205 +++ b/thesis/chapters/90_conclusion.tex
206 @@ -8 +8 @@
207 -as the lack of patterns and  predictability. However, it is not
    that easy to create a non-deterministic random bit generator
208 +as the lack of patterns and  predictability. However, it is very
    challenging to create a non-deterministic random bit generator
209 @@ -17 +17 @@
210 -However, it has been shown within Chapter~\ref{sec:results}, that
    it is possible to generate a sequence of random bits
211 +However, it has been shown in Chapter~\ref{sec:results}, that it is
     possible to generate a sequence of random bits
212 @@ -19 +19 @@
213 -yet unsolved issue occurred as described in Chapter~\ref{subsec:
    fail_on_other_devices}, when the raw material videos are gathered
214 +yet unsolved issue occurred as described in Chapter~\ref{subsec:
    fail_on_other_devices}, when the raw material videos are gathered
      %% are gattheres / tense or contition
215 @@ -23 +23 @@
216 -Master Thesis at hand but also has the downside of one failed test
    and a long duration of gathering the random numbers.\par
217 +Master Thesis but also has the downside of one failed test and a
    long duration of gathering the random numbers.\par
218 @@ -25,2 +25,2 @@
219 -Quantis QRNG does not always succeed all statistical tests.
    Moreover after the same binary output of the QRNG has been
220 -processed through the proposed algorithm within ERBG all
```

```
    statistical test have succeeded.
221 +Quantis QRNG does not always succeed all statistical tests.
    Moreover, all statistical test have succeeded after the same
222 +binary output of the QRNG has been processed through the proposed
    algorithm within ERBG.
223 @@ -37,2 +37,2 @@
224 -particle decays within their half-life. However, as literature and
    its thereof resulting smartphone application
225 -called ``CRAYFIS'' have shown, it is possible to detect single
    muons with the sensitive CMOS sensor of smartphones.\par
226 +particle decays within their half-life. However, it is possible to
    detect single muons with the sensitive CMOS sensor
227 +of smartphones as \cite{whiteson:2015}, has shown with the
    smartphone application called ``CRAYFIS'' in his scientific work
    .\par
228
229
230 diff --git a/thesis/thesis.pdf b/thesis/thesis.pdf
231 index a0fe074..42af395 100644
232 Binary files a/thesis/thesis.pdf and b/thesis/thesis.pdf differ
```

Listing B.2: Shows the exact change between digital submitted and printed version

Acronyms

BSI Backside Illuminated

CCD Charge Coupled Device

CERN Conseil Européen pour la Recherche Nucléaire

CMOS Complementary Metal Oxide Semiconductor

DRBG Deterministic Random Bit Generator

DRNG Deterministic Random Number Generator

ERBG Economical Random Bit Generator

ERNG Economical Random Number Generator

HDD Hard Drive Disc

NIST National Institute of Standards and Technology

NRBG Non-deterministic Random Bit Generator

PRNG Pseudo Random Number Generator

PBS Polarized Beam Splitter

QRNG Quantum Random Number Generator

RBG Random Bit Generator

RNG Random Number Generator

TRNG True Random Number Generator

Bibliography

AXIS Communications AB (2010). *CCD and CMOS sensor technology, Technical White Paper*. AXIS Communications AB. Available from: https://www.axis.com/files/whitepaper/wp_ccd_cmos_40722_en_1010_lo.pdf.

Barker, Elaine (2016). *Recommendation for Key Managment, NIST Special Publication 800-57 Part1 Revision 4*. 100 Bureau Drive, Gaithersburg, Maryland 20899, Unites States of America: National Institute of Technology, Computer Security Division. DOI: 10.6028/NIST.SP.800-57pt1r4.

Barker, Elaine and John Kelsey (2015). *Recommendation for Random Number Generation Using Deterministic Random Bit Generators , NIST Special Publication 800-90A Revision 1*. 100 Bureau Drive, Gaithersburg, Maryland 20899, Unites States of America: National Institute of Technology, Computer Security Division. DOI: 10.6028/NIST.SP.800-90Ar1.

– (2016). *Recommendation for Random Bit Generator (RBG) Constructions, NIST Special Publication 800-90C (Second Draft)*. 100 Bureau Drive, Gaithersburg, Maryland 20899, Unites States of America: National Institute of Technology, Computer Security Division. DOI: 10.6028/NIST.SP.800-90B.

Batanero, Carmen, David R. Green, and Luis Romero Serrano (1998). „Randomness, its meanings and educational implications“. In: *International Journal of Mathematical Education in Science and Technology* 29.1, pp. 113–123. DOI: 10.1080/0020739980290111.

Bayer, Bryce E. (1976). *Color Imaging Array. US-Patent.Nr.:3,971,065*. United States Patent Office. Available from: https://patentimages.storage.googleapis.com/89/c6/87/c4fb7fbb6d0a0d/US3971065.pdf.

Bennet, D.J. (1993). *The development of the mathematical concept of randomness*. New York Univeristy. ISBN: 9317657.

Bewick, Sharon, Richard Parsons, Therese Forsythe, Shonna Robinson, and Jean Dupon (2016). *Book: Introductory Chemistry (CK-12)*. Wikimedia Commons. Available from: https://chem.libretexts.org/Bookshelves/Introductory_Chemistry/Book%3A_Introductory_Chemistry_(CK-12) [Apr. 10, 2019].

Bierhorst, Peter, Emanuel Knill, Scott Glancy, Yanbao Zhang, Alan Mink, Stephen Jordan, Andrea Rommal, Yi-Kai Liu, Bradley Christensen, Sae Woo Nam, Martin J. Stevens, and Lynden K. Shalm (2018). „Experimentally generated randomness certified by the impossibility of superluminal signals“. In: *Nature* 556.7700, pp. 223–226. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0019-0.

Boyle, Willard Sterling and George Elwood Smith (1974). *Buried Channel CHarge Coupled Devices. US-Patent.Nr.:3,792,322*. United States Patent Office. Available from: https://patentimages.storage.googleapis.com/90/6e/28/e1f0a9b89d5110/US3792322.pdf.

Braunecker, Christoph (2012). *Pseudo-random number generation using deterministic chaotic systems Theoretical analysis and practical implementation*. FH JOANNEUM (University of Applied Sciences), Kapfenberg.

Burnett, Colin M.L. (2006a). *Profile/cross-section of sensor*. Wikimedia Commons. Available from: https://en.wikipedia.org/wiki/Bayer_filter#/media/File:Bayer_pattern_on_sensor_profile.svg [Apr. 12, 2019].

– (2006b). *The Bayer arrangement of color filters on the pixel array of an image sensor*. Wikimedia Commons. Available from: https://en.wikipedia.org/wiki/Bayer_filter#/media/File:Bayer_pattern_on_sensor.svg [Apr. 12, 2019].

Chang, Lanlan and Yap-Peng Tan (2006). *Hybrid color filter array demosaicking for effective artifact suppression*. Nanyang University, School of Electrical Engineering, Nanyang Avenue, 639798 Singapore. Available from: https://web.archive.org/web/20091229063407/http://www3.ntu.edu.sg/home5/CHAN0069/JEI013003.pdf [Apr. 12, 2019].

Chen, I-Te (2013). „Random Numbers Generated from Audio and Video Sources“. In: *Hindawi Publishing Corporation Mathematical Problems in Engineering* Volume 2013, Article ID 285373, 7 pages.

DeviceSpecifications.com (2019). *Comparison between LG Nexus 5X and Xiaomi Mi A1*. devicespecifications.com. Available from: https://www.devicespecifications.com/en/comparison/bf0ed3892.

Einstein, Albert (1926). *Brief von Albert Einstein an Max Born vom 04 Dezember 1926*.

Fraknoi, Andrew, David Morrison, and Sidney C. Wolff (2019). *Astronomy*. Rice University, Houston, Texas. ISBN: 978-1-938168-28-4.

Holmes, Susan, Persi Diaconis, and Richard Montgomery (2004). *Dynamical Bias in the Coin Toss*. Stanford University, Stanford, California, pp. 1–31.

Homann, Jan (2009). *Visible spectrum of hydrogen*. Available from: https://en.wikipedia.org/wiki/Balmer_series#/media/File:visible_spectrum_of_hydrogen.jpg.

Hopf, Eberhard (1934). „On Causality, Statistics and Probability“. In: *Journal of Mathematics and Physics* 17, pp. 51–102. DOI: 10.1002/sapm193413151.

– (1936). „Über die Bedeutung der willkürlichen Funktionen für die Wahrscheinlichkeitstheorie“. In: *Jahresbericht Deutsche Mathematik* 46, pp. 179–195.

– (1937). „Ein Verteilungsproblem bei dissipativen dynamischen Systemen“. In: *Mathematische Annalen* 114, pp. 161–186.

ID Quantique, Swiss Quantum (2016). *Quantis, When Random Numbers Cannot be left to chance*. ID Quantique SA, Chemin de la Marbrerie 3, Geneva, Switzerland. Available from: https://marketing.idquantique.com/acton/attachment/11868/f-021f/1/-/-/-/-/Quantis%20QRNG_Brochure.pdf.

Jaros, Albert, Alfred Nussbaumer, and Peter Nussbaumer (2012). *PHYSIK compact, Basiswissen 6 RG*. öbv. ISBN: 978-3-209-07212-2.

Jennewein, Thomas, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, and Anton Zeilinger (2000). „A fast and compact quantum random number generator“. In: *Review of Scientific Instruments* 71.4, pp. 1675–1680. DOI: 10.1063/1.1150518. Available from: https://arxiv.org/pdf/quant-ph/9912118.pdf.

Jeroen Hoerling, Door (2014). *Samsung NX1 Preview, Gericht op filmers en semiprofessionnals*. tweakers.net. Available from: https://tweakers.net/reviews/3708/2/samsung-nx1-gericht-op-filmers-en-semiprofessionals-bsi-cmos-sensor-met-28-megapixels.html.

Johnston, David (2018). *Random Number Generators - Principles and Practices: A Guide for Engineers and Programmers*. De|G Press. Available from: https://books.google.at/books?id=kIxuDwAAQBAJ&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q=underflow&f=false.

Krhovják, Jan, Petr Švenda, and Václav Matyáš (2007). „The Sources of Randomness in Mobile Devices“. In: Nordsec 2007: The 12th Nordic Conference on Secure IT Systems. Reykjavik University.

Leschiutta, Riccardo (2015). *PARAPPNOID: UN'APPLICAZIONE DI MESSAGGISTICA BASATA SU ONE-TIME PAD*. Received Thesis Paper through email conversation with Mr.Leschiutta. UNIVERSITA DEGLI STUDI DI UDINE, Dipartimento di Scienze Matematiche, Informatiche e Fisiche.

– (2016). *Java True Random Number Generator (TRNG) that uses JPEG images as entropy source.* prgpascal@Github.com. Available from: https://github.com/prgpascal/bluerand.

LG Electronics, Inc. (2015). *LG Nexus 5X*. Amazon.com, Inc. Available from: https://www.amazon.de/LG-Smartphone-Display-Android-Marshmallow-Carbon/dp/B015Y5398A/ [Apr. 11, 2019].

Mehl, James B. and Michael R. Moldover (1986). *Measurement of the ratio of the speed of sound to the speed of light*. 100 Bureau Drive, Gaithersburg, Maryland 20899, Unites States of America. DOI: 10.1103/PhysRevA.34.3341.

Mewaldt, Richard A. (2019). *Cosmic Rays*. This article was accepted for publication in the Macmillan Encyclopedia of Physics in 1996. It presents a general introduction to the field of cosmic rays. Available from: http://www.srl.caltech.edu/personnel/dick/cos_encyc.html [Jan. 27, 2019].

Poincare, Henri (1936). *Foundation of Science:Chance Translated by J.R. Newman*. The World of Mathematics.

Purves, Dale, George J Augustine, David Fitzpatrick, William C. Hall, Anthony-Samuel Lamantia, James O. McNamara, and Mark Williams (2004). *Neuroscience. 2nd edition*. Sinauer Associates Inc., Sunderland, Massachusetts 01375, United States of America. ISBN: 0-87893-725-0. Available from: https://www.ncbi.nlm.nih.gov/books/NBK10885/ [Apr. 10, 2019].

Roberts-Breslin, J. (2012). *Making Media: Foundations of Sound and Image Production*. Focal Press. ISBN: 9780240815275.

Robjohns, Hugh (2010). *A brief history of microphones*. Microphone Data Book.

Ronan, Philip (2019). *Electromagnetical Spectrum*. Wikimedia Commons. Available from: https://commons.wikimedia.org/wiki/File:EM_spectrum.svg [Apr. 8, 2019].

Rukhin, Andrew, Rukhin Soto, Jamesand Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo (2010). *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22 Revision 1*. 100 Bureau Drive, Gaithersburg, Maryland 20899, Unites States of America: National Institute of Technology, Computer Security Division and Statistical Engineering Divison. DOI: 10.6028/NIST.SP.800-22r1a.

Schiller, Jeffrey (2005). *Randomness Requirements for Security*. IETF Administration LLC.

Smith, H.E. (1999). *Gene Smith's Astronomy Tutorial: Stellar Spectra*. University of California, San Diego, La Jolla, California 92093-0424, United States of America. Available from: http://casswww.ucsd.edu/archive/public/tutorial/Stars.html [Apr. 8, 2019].

Suda, Martin (2015). „Course: Quantum Cryptography Fundamentals". Unpublished: Part of course scriptum used in lecture Cryptography within the lecture sub part Quantum Cryptography from the study IT & Mobile Security on the FH JOANNEUM, University of Applied Sciences, Kapfenberg.

Trobe, Jonathan (2014). Kellog Eye Center, University of Michigan Health System. Available from: http://kellogg.umich.edu/theeyeshaveit/anatomy/section-retina.html [Apr. 8, 2019].

Turan, Meltem Sönmez, Elaine Barker, John Kelsey, Kerry A. McKay, Mary L. Baish, and Mike Boyle (2018). *Recommendation for the Entropy Sources Used for Random Bit Generation, NIST Special Publication 800-90B*. 100 Bureau Drive, Gaithersburg, Maryland 20899, Unites States of America: National Institute of Technology, Computer Security Division. DOI: 10.6028/NIST.SP.800-90B.

Turchetta, Renato, Kenneth R. Spring, and Michael W. Davidson (2019). *Introducing to CMOS Image Sensors*. Olympus Scientific Solutions Corp. Available from: https://www.olympus-lifescience.com/en/microscope-resource/primer/digitalimaging/cmosimagesensors/.

Wanlass, Frank M (1967). *Low Stand-By Power Complementary Field Effect Circuitry. US-Patent.Nr.:3,356,858*. United States Patent Office. Available from: https://patentimages.storage.googleapis.com/a9/a5/6e/8688ff036edb08/US3356858.pdf.

Whiteson, Daniel, Michael Mulhearn, Chase Shimmin, Kyle Cranmer, Kyle Brodie, and Dustin Burns (2015). *Observing Ultra-High Energy Cosmic Rays with Smartphones*. arXiv, Cornell University.

Xiaomi (2017). *Xiaomi Mi A1*. Xiaomi. Available from: https://www.mi.com/global/mi-a1/specs/ [Apr. 11, 2019].

Zhang, Xuping, Li Qi, Zhiqiang Tang, and Yixin Zhang (2012). „Portable true random number generator for personal encryption application based on smartphone camera". In: *Electronic Letters* 50, pp. 1841–1843.