

**MASTER**

**Acoustical physical uncloneable functions**

Vrijaldenhoven, S.C.B.J.

*Award date:*  
2004

[Link to publication](#)

**Disclaimer**

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

**TECHNISCHE UNIVERSITEIT EINDHOVEN**

Department of Mathematics and Computing Science

MASTER'S THESIS

**Acoustical Physical Uncloneable Functions**

by

Serge Vrijaldenhoven

**THIS REPORT IS CONFIDENTIAL**

Supervisors :

dr. R.M. Aarts — Philips Research, DSP-acoustics & Sound Reproduction  
dr. C. Huizing — TU/e, Department of Mathematics and Computer Science  
Computational Engineering - Visualization  
dr. E.P. de Vink — TU/e, Department of Mathematics and Computer Science  
Software Technology - Formal Methods

Philips Research  
Prof. Holstlaan 4  
5656 AA Eindhoven  
The Netherlands  
+31(0)40 279 1111  
<http://www.research.philips.com>

Eindhoven University of Technology  
Den Dolech 2, Postbus 513  
5600 MB Eindhoven  
The Netherlands  
+31(0)40 247 9111  
<http://www.tue.nl>

Eindhoven, October, 2004



---

## Acoustical Physical Uncloneable Functions — Confidential Abstract

by Serge Vrijaldenhoven

Today's cryptography extensively makes use of one-way functions: functions that are easy to compute but difficult to invert. During the last years interest has risen for *Physical One-Way Functions*: physical systems, rather than mathematical ones, that have this property. If such a system has some random factor in the production process, then the response of each system to certain inputs distinguishes the system from others. Because of their unique responses these systems are often called *Physical Uncloneable Functions (PUFs)*. Cloning here refers to making a physical copy or constructing a simulation model that predicts the behavior of the system. During the last years much research has been devoted to Silicone PUFs and Optical PUFs. Philips is interested in another type of PUF that has not been under research so far (as far that is known): the Acoustical PUF (APUF). Experiments with delay lines (the DL701 - used for delaying signals in old-fashioned tv's) are used to evaluate the feasibility of creating APUFs.

PUFs can be deployed in many cryptographic applications. Every application that makes use of a 'securely stored' random key could use a PUF instead. The advantage of using a PUF is that the PUF cannot be physically copied - not even by the manufacturer. The goal is to create a PUF together with a very large amount of challenge-response pairs (CRPs). This means that even brute force (attacks by measuring and storing all or all relevant CRPs) or dictionary attacks become unfeasible when the time to measure one CRP is substantial. Every time the key is used another CRP is involved to prevent replay attacks.

The DL701s should be identifiable: if there is not enough differentiation from the noisy measurements, then the objects cannot be used as PUFs. The obtained result is that by using principle component analysis (PCA) to extract important features, the DL701s can be differentiated. If a False Rejection Rate is allowed of 1 out of 10,000 a False Acceptance Rate of 1 out of 100,000 should be easily possible, especially if temperature control will be used.

It is important to know how many unique APUFs can be produced. Just like normal iron keys (typically  $10^6$  keys possible for one type of lock), the more unique keys, the less chance that someone has the same key. The upper bound for the number of unique DL701s that (theoretically) can be created is estimated to be  $2^{28 \cdot 10^3}$ . APUFs with different parameters can be created which probably enlarges this number.

A demo system (called AKI — **A**PUF **K**ey extraction and **I**dentification system) that uses DL701s has been created. It has different modes of operation. It can identify an enrolled DL701 automatically when it is placed in the system and extract keys in two different modes. First method is based on a threshold. This extraction method proved unsatisfactory since the bits that are extracted seem to be correlated and only a few effective bits result. The second method is based on quantization. This method is promising since likely many bits are independent (45 to 240 bits of the 60 components extracted (depending on the number of quantization values)).

In the end the user could use the key extracted with AKI to encrypt/decrypt files or messages or just like any other cryptographic key.



## Physical Uncloneable Functions — Non-confidential Abstract

by Serge Vrijaldenhoven

Submitted to the Department of Mathematics and Computing Science of the Eindhoven University of Technology on October 2004

Today's cryptography extensively makes use of one-way functions: functions that are easy to compute but difficult to invert. During the last years interest has risen for *Physical One-Way Functions*: physical systems, rather than mathematical ones, that have this property. If such a system has some random factor in the production process, then the response of each system to certain inputs distinguishes the system from others. Because of their unique responses these systems are often called *Physical Uncloneable Functions (PUFs)*. Cloning here refers to making a physical copy or constructing a simulation model that predicts the behavior of the system. During the last years much research has been devoted to Silicone PUFs and Optical PUFs. Philips is interested in another type of PUF that has not been under research so far (as far that is known): let us call this the Researched PUF (RPUF). Experiments with such type of objects (called Rs) are used to evaluate the feasibility of creating RPUFs.

PUFs can be deployed in many cryptographic applications. Every application that makes use of a 'securely stored' random key could use a PUF instead. The advantage of using a PUF is that the PUF cannot be physically copied - not even by the manufacturer. The goal is to create a PUF together with a very large amount of challenge-response pairs (CRPs). This means that even brute force (attacks by measuring and storing all or all relevant CRPs) or dictionary attacks become unfeasible when the time to measure one CRP is substantial. Every time the key is used another CRP is involved to prevent replay attacks.

The Rs should be identifiable: if there is not enough differentiation from the noisy measurements, then the objects cannot be used as PUFs. The obtained result is that by using principle component analysis (PCA) to extract important features, the Rs can be differentiated. If a False Rejection Rate is allowed of 1 out of 10,000 a False Acceptance Rate of 1 out of 100,000 should be easily possible, especially if temperature control will be used. The upper bound for the number of unique Rs that (theoretically) can be created is estimated to be  $2^{28 \cdot 10^3}$ . RPUFs with different parameters can be created which probably enlarges this number.

A demo system (called RKI — **R**PUF **K**ey extraction and **I**dentification system) that uses Rs has been created. It has different modes of operation. It can identify an enrolled Rs automatically when it is placed in the system and extract keys in two different modes. First method is based on a threshold. This extraction method proved unsatisfactory since the bits that are extracted seem to be correlated and only a few effective bits result. The second method is based on quantization. This method is promising since many bits seem to be independent (45 to 240 bits of the 60 components extracted (depending on the number of quantization values)).

In the end the user could use the key extracted with RKI to encrypt/decrypt files or messages or just like any other cryptographic key.

Supervisors:

dr. R.M. Aarts  
dr. C. Huizing  
dr. E.P. de Vink

—  
—  
—

Philips Research, DSP-Acoustics & Sound Reproduction  
TU/e, Department of Mathematics and Computer Science  
Computational Engineering - Visualization  
TU/e, Department of Mathematics and Computer Science  
Software Technology - Formal Methods



# Acknowledgements

I like to keep things simple, very simple

I thank all people close and around me  
— you feel it if you are...





# Contents

<b>1. Theory of PUFs</b>	<b>3</b>
1.1. General	3
1.2. Challenge-Response Pairs	5
1.3. Attacks on PUFs	5
1.3.1. Duplication	5
1.3.2. Exhaustive Model Building	6
1.3.3. Adaptive Model Building	6
1.4. Controlled PUFs	6
1.5. Applications of PUFs	8
1.5.1. Example Application: PUFs replace TAN lists	9
1.5.2. Example Application: PUFs on SIMs	10
1.6. Key Generating Schemes	11
1.6.1. Mean Threshold	12
1.6.2. Quantized Key Extraction	14
<b>2. Glass Delay Line as APUF</b>	<b>25</b>
2.1. Delay Lines in General	25
2.2. The DL701	27
2.3. Sound Propagation Through Solids	28
2.4. DL701 as APUF	29
2.5. Number of Possible Structures	30
<b>3. System Setup</b>	<b>33</b>
3.1. Setup	33
3.2. Communication	34
<b>4. Experiments and Results</b>	<b>39</b>
4.1. Typical Measurement	39
4.1.1. Characteristic Frequency Spectrum	39
4.2. Identification of DL701s	41
4.2.1. Entropy of the Principle Components	43
4.2.2. Demo System DL701 Identification	47
4.2.3. Conclusion	50
4.3. Threshold Key Generation	50
4.3.1. Conclusion	52
4.4. Quantized Key Generation	52
4.4.1. Conclusion	57
4.5. PUF on chip Experiment	57

<b>5. Conclusion</b>	<b>59</b>
5.1. Future work . . . . .	59
<b>A. Cryptography</b>	<b>63</b>
A.1. Symmetric Encryption Algorithms . . . . .	63
A.1.1. One-time pad . . . . .	63
A.1.2. Stream/Block Ciphers . . . . .	63
A.2. Perfect Secrecy . . . . .	63
A.3. Hash and one-way functions . . . . .	65
A.4. Asymmetric Encryption . . . . .	66
A.5. Digital Signatures . . . . .	67
A.6. Identification of Users . . . . .	67
A.7. (Pseudo) Random Number Generators . . . . .	68
A.7.1. Attacks on PRNGs . . . . .	69
A.7.2. Skew Correction . . . . .	69
A.8. Entropy . . . . .	71
A.8.1. Bits of Information . . . . .	72
A.8.2. Entropy of a measurement value . . . . .	72
<b>B. Plotting Functions</b>	<b>75</b>
B.1. Plotting $q \cdot f_W(w s = j)$ . . . . .	75
B.2. Plotting $I(W; S)$ . . . . .	75
<b>C. AKI Manual</b>	<b>79</b>
C.1. PCI-GPIB Card Installation . . . . .	79
C.2. Starting the System . . . . .	80
C.2.1. Starting the Network Analyzer . . . . .	80
C.2.2. Preparing the Analyzer . . . . .	80
C.2.3. Sweep Time . . . . .	81
C.2.4. Plotting and Conversion of the Network Analyzer Values . . . . .	81
C.2.5. Starting AKI . . . . .	82
C.3. Auto Identification . . . . .	83
C.3.1. Enrollment for Auto Identification . . . . .	83
C.3.2. Auto Identification . . . . .	84
C.4. Threshold Key Extraction . . . . .	85
C.4.1. Enrollment . . . . .	85
C.4.2. Key Extraction . . . . .	86
C.5. Quantized Key Extraction . . . . .	87
C.5.1. Enrollment . . . . .	88
C.5.2. Key Extraction . . . . .	95



# 1. Theory of PUFs

This chapter discusses various characteristics and definitions of PUFs. What exactly are PUFs? How do they fit in cryptographic applications and how can keys be extracted from them? People not familiar with cryptography may want to consult section A in the appendix first. It is a short introduction in cryptography and also contains information about the entropy of measurement values.

## 1.1. General

PUFs can be used for all kinds of fields related to cryptography and security. This section starts off with some definitions about PUFs as given in [16].

**Definition 1** *A Physical Uncloneable Function (PUF) is a function that maps challenges to responses, that is realized by a physical system, and verifies the following properties:*

- *Easy to evaluate: The physical device is easily capable of evaluating the function in a short amount of time.*
- *Hard to characterize: From a polynomial number of plausible physical measurements (in particular, determination of chosen challenge-response pairs (CRPs) ), an attacker who no longer has the device, and who can only use a polynomial amount of resources (time, matter, etc. . . ) can only extract a negligible amount of information about the response to a randomly chosen challenge.*

Short and polynomial are relative to the size of the device where short means linear or low degree polynomial. Plausible is relative to the current state of the art measurement techniques and will likely change when improved methods are devised.

**Definition 2** *A type of PUF is said to be Manufacturer Resistant if it is technically impossible to produce two identical PUFs of this type given only a polynomial amount of resources (time, money, silicon, etc.).*

Of course manufacturer resistant PUFs are the most interesting, since they can be used to make unique systems that cannot be reproduced, not even by the manufacturer.

**Definition 3** *A PUF is said to be Controlled (CPUF) if it can only be accessed via an algorithm that is physically linked to the PUF in an inseparable way (i.e., any attempt to circumvent the algorithm will lead to the destruction of the CPUF). In particular this algorithm can restrict challenges that are presented to the PUF and can limit the information about responses that is given to the outside world.*

In practice, linking an algorithm to a PUF in an inseparable way is quite difficult. However, it is still easier to do than to link an algorithm to a digital secret key in an inseparable way, which is what current smartcard technology attempts.

The physical system is designed such that it interacts in a complicated way with stimuli (challenges) and leads to unique but unpredictable responses. Hence a PUF can be compared with a keyed hash function (see section A.3 on page 65). The key is the physical system consisting of many ‘random’ components. In order to be hard to characterize, it should be difficult to extract the relevant properties of the interacting components by measurements from the system. If the physical function is based on many complex interactions, then mathematical modelling is hard. Furthermore it should be hard to produce a physical copy of the PUF. Good candidates are physical systems that are produced by production processes that contain (measurable) randomness. Some examples of PUFs are digital PUFs, optical PUFs, silicon PUFs and acoustical PUFs.

**Digital PUFs** A tamper resistant environment protects a secret key. The secret key is used for encryption and authentication purposes. (Note that a physical copy can easily be made by the manufacturer in this case).

Currently smartcards link a conventional digital secret key to an algorithm in an inseparable way. Barriers are incorporated to protect the secret key. Suppose an integrated circuit (IC) is equipped with a secret key  $k$ , a hash function  $h$  and tamper resistant technology is used to protect  $k$ . Then the following function is a PUF

$$x \rightarrow h(k, x) \tag{1.1}$$

This kind of PUF is not always satisfactory for several reasons:

- High quality tamper resistant technology is needed, which is expensive and difficult to create.
- The digital PUF is not tamper manufacturer resistant. The PUF creator can create multiple ICs with the same secret key.

Because of these two weaknesses, a digital PUF does not offer any security advantage over storing a key in digital form, and hence it is cheaper to use a conventional key storage system (without tamper resistant environment).

**Optical PUFs** Optical structures containing some scattering material are irradiated with coherent light. The speckle patterns that result from this scattering are unique. The challenge can be e.g. the angle of incidence, focal distance or wavelength of the light, a mask pattern blocking part of the laser light, or any other change in the wave front.

Modelling of optical PUFs is difficult [26] due to the inherent complexity of multiple coherent scattering. Even when the details of all scatterers are known, computation of the response using Feynman diagrams requires summation over a number of diagrams that grows exponentially with the number of scattering events.

Physically copying is difficult for two reasons: (i) The light diffusion obscures the locations of scatterers. Currently the best physical techniques can probe diffusive materials up to a depth of  $\approx 10$  scattering lengths. (ii) Positioning of a large number of scatterers is time consuming and requires a production process different from the original randomized process.

**Silicone PUFs** Manufacturing variation (properties of logical gates) between chips is used to produce unique responses. When these PUFs are probed at frequencies that are out of specs, a unique, unpredictable response is obtained in the form of delay times.

**Acoustical PUFs** Acoustical PUFs (APUFs) are a new idea (as far that is known) proposed in this thesis. An electrical signal (oscillating voltage) is transformed to an identical mechanical vibration by a transducer. This vibration propagates as a sound wave through a solid medium, which contains scatterers (for the material is inhomogeneous). Finally the wave arrives at another transducer which converts the wave to an electrical signal again. The signals that result from this scattering are unique for each APUF. The manufacturing variation between small plates of some kind of material (e.g. glass) is used to create unique systems.

## 1.2. Challenge-Response Pairs

Given a PUF, challenge and response pairs (CRPs) can be generated. The challenge can be an input stimulus and the response depends on the transient behavior of the PUF. A number of different challenges can be used to generate the CRPs. The number of bits that can be extracted for one challenge is up to

$$\frac{1}{2} \log_2 \left( 1 + \frac{\sigma_{inter}^2}{\sigma_{noise}^2} \right) \quad (1.2)$$

(see equation A.24 on page 73) where  $\sigma_{inter}$  and  $\sigma_{noise}$  are the standard deviation of inter-PUF variation and the measurement noise respectively. When one challenge does not provide a large enough number of identification bits to authenticate a PUF, multiple independent challenges may be used. So when a system containing a PUF is authenticated, a set of CRPs is potentially revealed to an adversary. In order to prevent replay attacks the same CRPs cannot be used again. If the adversary can learn the whole set of CRPs, she can create a model of a counterfeit PUF. A database of CRPs has to be maintained by the entity that wishes to authenticate the PUF. The database only needs to store a small (random) subset of all CRPs possible provided that when the database runs out of CRPs it has to be recharged again. This can be done by going to the entity that performs the authentication. Of course the database has to be kept secret so the attacker is unable to predict the challenges. With controlled PUFs (see section 1.4 on the following page) a lot of these limitations can be overcome. For example the reuse of a CRP can be considered and storing new CRPs can be done over an untrusted network.

## 1.3. Attacks on PUFs

There are several possible attacks on PUFs: duplication, model building using direct measurement, model building using adaptively-chosen challenge generation and violation of controlled PUF's control mechanism. In this section these attacks will be discussed shortly.

### 1.3.1. Duplication

An adversary can attempt to physically create a PUF that has the same characteristics. But due to statistical variation, unless the PUF is very simple, the adversary will have to create

a large number of PUFs and precisely characterize each one of them in order to create and discover a counterfeit.

### 1.3.2. Exhaustive Model Building

An adversary that has unrestricted access to the PUF can attempt to create a model of the PUF by measuring the responses to all possible challenges. Note that a brute force attack tries to break the system by *guessing* responses (usually on the reader system side).

### 1.3.3. Adaptive Model Building

The adversary could try to build a model of the PUF by measuring the response of the PUF to a polynomial number of adaptively chosen challenges. The number of physical parameters that define a PUF is proportional to the size of the system that defines it. So when an adversary is able to determine some primitive parameters that are proportional to the size of the system, she can use them to simulate the system. To determine these primitive parameters, the adversary could measure a number of CRPs and use them to build a system of equations that she can try to solve (for an example on silicone PUFs see [17]). For a real PUF these equations should be impossible to solve with a polynomial amount of resources (by definition of a PUF). However, there can be physical systems for which most CRPs lead to unsolvable equations, but a small subset of CRPs do give equations that are able to break the PUF (hence this is not a real PUF). Consequently such a system is not secure: an attacker can use these CRPs to get a solvable system of equations, calculate the primitive parameters, and clone the PUF by building a simulation model. This is probably the most plausible form of attack. More research to modelling of piezoelectric transducers and waves in solids should be done in order to investigate how hard this is.

## 1.4. Controlled PUFs

Using control as described in [16] it is possible to make a PUF more robust and reliable. Figure 1.1 shows how a PUF can be controlled to improve it. The control layer can for example refuse challenges that lead to simple equations. But unfortunately the methods the adversary can use to get a simple set of equations from chosen CRPs is unknown beforehand. In addition to attacking a PUF directly an adversary can attempt to violate a controlled PUF's

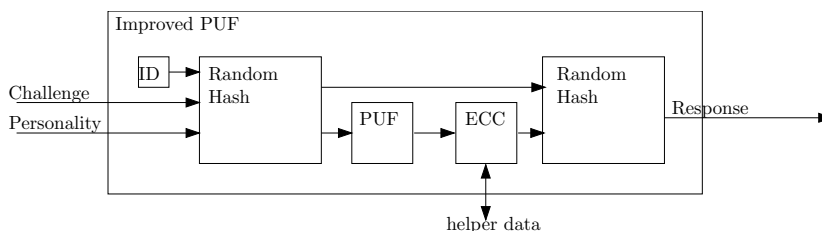


Figure 1.1.: Using control to improve a PUF.

control mechanism. This includes trying to get direct access to the PUF, or trying to violate the control algorithm. The best way to avoid this kind of attack is for the algorithm to be embedded within the physical system that defines the PUF. In order to get to the algorithm



the adversary has to damage the PUF, which renders the attack useless. To prevent the adversary from performing a chosen challenge attack on the PUF (used in adaptive model building, see section 1.3.3) a hash function is placed before the PUF. Instead of using PUF  $f$  directly, a new PUF  $g$  is now constructed with:  $g(x) = f(h_1(x))$ , where  $h_1$  is the hash function. This prevents a model-building adversary from selecting challenges that allow him to extract parameters more easily, since  $h_1(x)$  cannot be chosen directly. Hence the designer of the PUF does not have to know what challenges the attacker might try to exploit.

The output of a physical system is likely to produce similar responses when faced with similar challenges. To de-correlate the final response from the physical measurements a hash function is placed after the PUF. The hash function's avalanche effect (see section A.3 on page 65) ensures that nearby lying physical responses lead to seemingly unrelated final outputs. This is expressed as:  $k(x) = h_2(x, f(x))$ . To set up a system of simple equations, the attacker now has to invert  $h_2$ . This makes model-building even harder. Placing a random function after the PUF makes the system provably resistant to modelling attacks, as long as enough information is extracted from the physical part before running it through the output random function.

A PUF with added control is called a controlled PUF or CPUF.

### Personality number

Since PUFs are unique they can be tracked. A growing number of people feels uncomfortable with the idea that they can be tracked directly or indirectly (by a PUF for example). In relation to PUFs this concern can be eliminated by creating a CPUF with multiple personalities. This means the owner of the CPUF has some parameter that she can control to show different 'faces' of her CPUF to different applications. This is realized by hashing the challenge with a personality number, and using that hash as the input to the rest of the CPUF. Now the owner effectively has many different PUFs and hence different third parties that interact with the CPUF cannot determine if they interacted with the same CPUF.

### Error correction

In most cases a response from a PUF is analog. Inevitable this means there will be noise on the PUF's output, which causes slight changes in the digitized output of the CPUF. In some applications it is desirable that the CPUF always has the exact same output for a certain challenge (for example when using a CPUF response to generate a key). One solution for this is to equip the PUF with an error correction algorithm. Together with a CRP also some error correcting code (ECC) is produced. When the PUF is challenged this redundant information (helperdata) is also provided and it is used to correct the direct response from the physical system. Of course it is very important that the error correcting code does not give away all the bits of the response.

### Unique identifier

Although a PUF can be manufacturer resistant, it could happen that two PUFs are the same. That is no problem because finding a pair of PUFs that is identical requires producing and comparing an unreasonable number of PUFs. To guarantee that two PUFs are different it is possible to give every CPUF a unique identifier (ID). When two CPUFs now have the same underlying PUF they are still different for the outside world, since the challenge will

be merged with the ID before running it through the rest of the system. The unique ID does not have to be secret. Of course the manufacturer can still find out two PUFs are the same before setting the ID, but the cost of testing for equality is very costly anyway.

To add more complexity to the adversary's problem it is possible to use the PUF system multiple times to produce one response. The response from one round can be feed back into the system and after a few rounds all the responses could be merged, together with the challenge, personality, and ID, and passed through the a hash function to produce the global response.

### Combinations of challenges

The PUF inside a CPUF can have much less CRPs then an uncontrolled PUF. This is because different combinations of the challenges for the PUF can be selected and seen as a new challenge for the controlled PUF. Suppose a controlled PUF is built with a PUF that has only 128 different 'measurement points' (each point is a CRP of the inner PUF). Then the CPUF challenge can be built from the 128 points by making  $N$  measurements to the inner PUF. Suppose all these 128 points have a different value. The challenge for the PUF inside the CPUF would be a random selection of  $N$  of these points and  $N \cdot \frac{\log 128}{\log 2}$  input bits are needed for the selection (points can be used more than once, selecting one point costs  $\frac{\log 128}{\log 2}$ ). This selection is determined by the output of the first hash function. Since all points have a different value, all selections can be distinguished, which effectively means the CPUF has  $128^N$  (!!!) different CRPs. However, usually the measurement values will only have 2 different values, reducing the number of different responses to  $2^N$ . It should not be possible to determine the outcome of the first hash function. This can be realized by also using some predefined part of the PUF as input to the hash function or by feeding back the first result from the PUF (not shown in the figure).

So for  $N = 128$  the CPUF system can generate  $2^{128} \approx 10^{38}$  different responses from a inner PUF with only 128 CRPs. This might make clear that a controlled PUF is substantially much stronger than an uncontrolled PUF. Note that the 128 measurement values of the inner PUF should be distributed evenly between 0 and 1.

The most important point of control is that with it, a PUF can be used to provide a shared secret to an application. Note however that the measurement on the PUF now takes place *inside* the CPUF self.

## 1.5. Applications of PUFs

PUFs can be used in a range of security and cryptographic related applications. They can be used for identification, authentication and key generation. They can be embedded into objects, such as smartcards, credit cards, the optics of a security camera, etc., preferably in an inseparable way (meaning the PUF gets damaged when removed). This makes the object in which a PUF is embedded uniquely identifiable and uncloneable. For some of the above mentioned applications control has to be added to the PUF (see section 1.4 on page 6).

### Authenticated identification

The most obvious application for PUFs is authentication (verified identification, i.e. making

sure an individual is who she says she is). It could for example be used to securely identify smartcards (plastic cards with an integrated circuit embedded in it). Each time the smartcard is presented to the authority the card reader asks the card for a response to a certain challenge. Each time the smartcard is presented again, different challenges are used in order to prevent replay attacks.

### **Proof of execution on a specific processor**

With chip authentication it is possible to prove that a specific computation was carried out on a certain chip. If chips can be authenticated, people that request computation can rely on the trustworthiness of the chip manufacturer who can substantiate that she produced the chip, instead of relying on the owner of the chip. Computation could be done on the secure chip or could be done on a faster system that is monitored by supervisory code on the secure chip.

### **Specific processor code**

The software industry is always looking for methods that can limit the use of their products. One could imagine a system where certain code could be adjusted so it runs only on a processor with a certain PUF in it. In this way, pirated code would fail to run on the system. A method could be to encrypt the code using the PUF's CRPs.

### **Encryption key**

If a bit-string is generated from the responses, this bit-string could be used for encryption and decryption of sensitive information. Of course this means the generated string must be the same each time the PUF is used. [15] describes in detail the protocols how to use a CRP of a controlled PUF to get to a shared secret between two parties.

#### **1.5.1. Example Application: PUFs replace TAN lists**

Currently the Dutch Bank 'Postbank' uses so-called TAN lists (Transaction Acceptation Number) to provide extra security measures for internet banking. The current system works as follows

- Customers apply for an internet banking account and are given an username and password.
- Additionally the customer:
  - is given a TAN list of 100 numbers (on paper), or
  - provides her mobile phone number.
- Customers that want to do some internet banking, login on an internet webpage by providing their username and password.
- When they instruct the Postbank to do a transaction, they are asked to give the TAN that matches with a certain serial number. TANs are only used once.
- The customer fills in the corresponding TAN from the list, or the TAN is sent by SMS to the customers mobile phone.

- If the correct TAN is provided, the transaction will be executed by the Postbank.

The TAN list system can be viewed as a challenge-response system, where the serial number is the challenge and the provided TAN is the response. The TAN list has to be kept in a safe place at all times. An attack to the TAN list system can go quite unnoticed. First the attacker has to obtain the users login name and password (which is said to be not too difficult with the average household PC, that on average lacks decent security measures). Then the attacker has to obtain the TAN list. This of course can be done by stealing it, but the user will notice that and block the internet account. Better is to copy the list. Although more difficult but much less intrusive is intercepting the SMS message (see section 1.5.2). This system can be made much safer by using PUFs for the CRPs

- TAN lists can be copied, PUFs cannot.
- TAN list are inconvenient when the numbers need to be longer (more secure). PUFs don't care about long numbers.

### 1.5.2. Example Application: PUFs on SIMs

The master key of a SIM (Subscriber Identity Module) in a mobile phone can be discovered, which makes SIM cloning possible [12]. Knowledge of the master key is sufficient to make calls billed to the customer that the SIM belongs to. Although meanwhile the GSM industry either uses proprietary algorithms or has replaced the standard cryptographic algorithm with a new one (COMP128v2/3 - which is not public, but 'available to GSM network operators and manufacturers of eligible GSM equipment' [5]), it is not unlikely that this algorithm will also be broken. One way or the other: the unique key is stored on the smartcard which makes it vulnerable even when it is protected. In short, the current system works as follows (see figure 1.2 on the facing page)

- Network challenges the SIM with  $RAND$ .
- The SIM computes a 32-bit long response  $SRES = A3(K_i, RAND)$  and sends this to the network ( $K_i$  is the master (shared) key of the SIM with the network,  $A3$  is an authentication algorithm).
- The network checks whether  $SRES$  is the same as the value it calculated itself.

A session key  $K_c$  is generated from the master key as follows:  $K_c = A8(K_i, RAND)$  ( $A8$  is a key agreement algorithm). This session key is used to initialize a stream cipher  $A5$  (see section A.1.2 on page 63) and the stream from the stream cipher algorithm is used to protect further messages. Every time when a session is initiated (voice or GPRS (General Packet Radio Service)), the same steps are executed, but with a different  $RAND$ . Several options with PUFs are possible to replace this system:

- One specific response of the PUF could replace the master key and the operator should learn this key from the PUF. Rest of the system stays the same. Attacking the SIM by opening it will now break the PUF and hence the key. However, when a bad authentication algorithm is used, the attack described in [12] is still possible. This replacement does not use the full potential of the PUF which makes (digital) cloning possible. It only protects against physical SIM probing attacks.

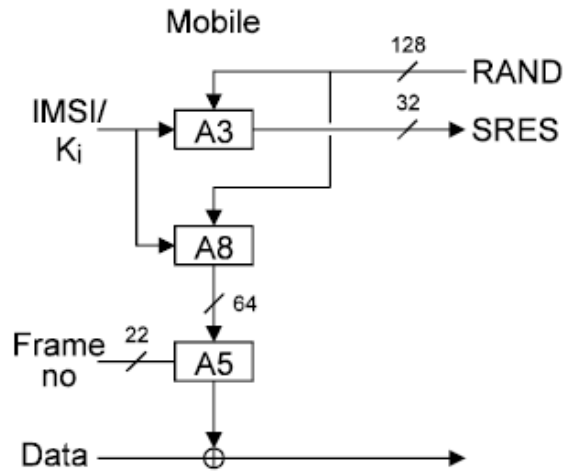


Figure 1.2.: The GSM protocol.

- Each time authentication is performed a different CRP of the PUF is used. Responses can be sent in clear since CRPs are used only once. If the CR-space is large enough, cloning is not possible anymore. When a mobile is authenticated, the network sends the challenge of the response that should be used as session key. Note that if the mobile phone supports a weak stream cipher (such as A5/2), the session key can be found [11] and the attacker is able to listen in on the conversation and see the data sent between the mobile and the network.

Of course there is a whole lot of other applications where PUFs can be useful, only imagination limits the possibilities of PUFs. It would be nice to have CPUFs inside mobile devices, so customers can use their mobile device for all kinds of cryptographic applications. The user could choose a different personality number for each application, so in fact only has to carry one CPUF around. This however does demand that every supplier of services used with a PUF, trust the PUF to be secure.

## 1.6. Key Generating Schemes

A fundamental property of a lot of cryptographic functions is that they are very sensitive to small disturbances in their inputs. Input data for these functions should not be noisy, which means that when measuring a physical system, some additional processing has to be performed to remove this noise. As already mentioned in section 1.4 on page 6 it is desirable that the PUF always has the exact same output for a certain challenge when it is used to generate a key. This sections describes two methods that can be used to get the desired result. Consider the schematic system in figure 1.3. Two phases are distinguished:

- Enrollment phase. During the enrollment phase properties of the object are measured (several times to reduce noise) with specialized equipment. From the measurement data and chosen secret  $\mathbf{S}$ , helper data is derived. The reference data stored in the database is obtained by applying a (possibly) one-way function  $h$  to  $\mathbf{S}$ .
- Recognition phase. When an object is used as a PUF later, the user inserts the PUF and supplies the ID of the PUF to the system. A (noisy) measurement is performed on

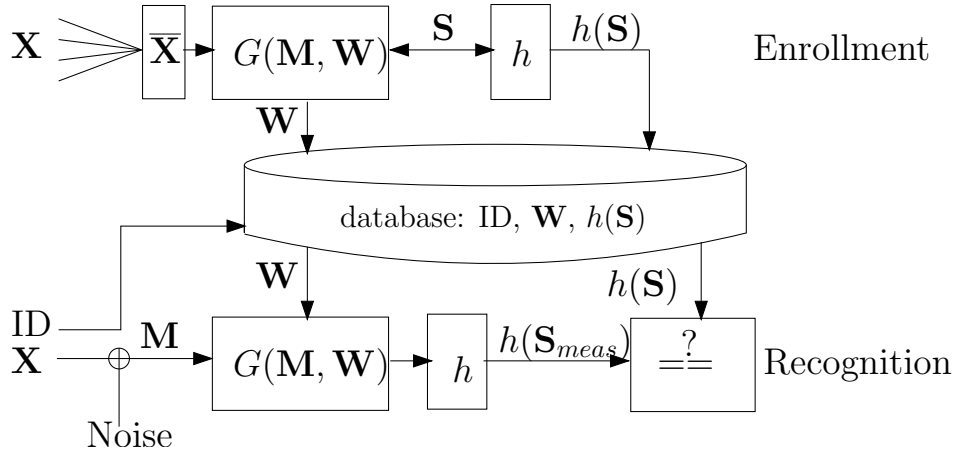


Figure 1.3.: Schematic presentation of PUF system.

$\mathbf{X}$  : Physiological properties of object ( $\bar{\mathbf{X}}$  the average of several measurements)

$\mathbf{M} = \mathbf{X} + \mathbf{N}$  : (possibly noisy) Measurement

ID : Unique ID that identifies the PUF

$\mathbf{S}$  : Chosen secret

$G(\mathbf{M}, \mathbf{W})$  : Signal processing function

$\mathbf{S}_{meas}$  : Estimate of the secret  $\mathbf{S}$

$\mathbf{W}$  : Helper data

Note that all bold printed symbols are vectors of dimension  $l$ .

the PUF and the helper data  $\mathbf{W}$  is communicated to the PUF. The helper data and measurement data are then processed by signal processing function  $G$  to construct a secret  $\mathbf{S}_{meas}$ . Finally,  $h(\mathbf{S}_{meas})$  is computed and compared to the stored data  $h(\mathbf{S})$  in the database.

PUF systems usually work like the system described above. They typically differ in the description of function  $G$ .

### 1.6.1. Mean Threshold

By taking a number of measurement signals from different objects and taking their mean, a threshold can be created. This threshold can be used to generate a bit-string with the same length  $l$  as the number of points in the measurement signal. The idea is very simple:

$$\mathbf{S}_{meas,i} = \begin{cases} 0 & \text{if } \mathbf{M}_i \geq \mathbf{T}_i \\ 1 & \text{if } \mathbf{M}_i < \mathbf{T}_i \end{cases} \quad (1.3)$$

with:

$i$  :  $i$ -th coordinate of vector,  $0 < i \leq l$

$\mathbf{M}_i$  : Measurement value

$\mathbf{T}_i$  : Non-ID based helper data which is actually a threshold value

$\mathbf{S}_{meas,i}$  : Derived secret value ( $S \in \{0, 1\}^l$ )

However, the  $\mathbf{S}_{meas,i}$ 's that belong to values of  $\mathbf{M}_i$  that lie very close to the threshold  $\mathbf{T}_i$  have a high chance of 'flipping' between 0 and 1 with each measurement. To get rid of these 'unsteady' bits, a list of bit numbers is generated during enrollment. This list contains the

bit numbers that flip during the enrollment measurements and hence are unstable. For each object this list is saved (helper data  $\mathbf{W}$ ).

If the unsteady bit list is determined accurately, this method results in a bit-string that is steady and hence can be used as an input for cryptographic functions. Practice however learns that new unsteady bits will appear due to changing environment variables, that did not change during the enrollment phase. To solve this a Bose Chaudhuri Hocquenghem (BCH) error correcting code [6, 3] can be used (see figure 1.4) together with a randomly generated secret to add confusion. BCH coding can encode a message into a (larger) codeword, when

N	K	T	N	K	T	N	K	T
7	4	1	255	199	7	511	358	18
15	11	1		191	8		349	19
	7	1		187	9		340	20
	5	1		179	10		331	21
31	26	1		171	11		322	22
	21	1		163	11		313	23
	16	1		155	11		304	24
	11	1		147	11		295	25
	6	1		139	11		286	26
63	57	1		131	11		277	27
	51	1		123	11		268	28
	45	1		115	11		259	29
	39	1		107	11		250	30
	36	1		99	11		241	31
	30	1		91	11		238	32
	24	1		87	11		229	33
	18	1		79	11		220	34
	16	1		71	11		211	35
	10	1		63	11		202	36
	7	1		55	11		193	37
127	120	1		47	11		184	38
	113	1		45	11		175	39
	106	1		43	11		166	40
	99	1		37	11		157	41
	92	1		39	11		148	42
	85	1		21	11		139	43
	78	1		13	11		130	44
	71	1		9	11		121	45
	64	1	511	502	11		112	46
	57	1		493	11		103	47
	50	1		484	11		94	48
	43	1		475	11		85	49
	36	1		466	11		76	50
	29	1		457	11		67	51
	22	1		448	11		58	52
	15	1		439	11		49	53
	8	1		430	11		40	54
255	247	1		421	11		31	55
	239	1		412	11		28	56
	231	1		403	11		19	57
	223	1		394	11		10	58
	215	1		385	11			59
	207	1		376	11			60
				367	11			61

N: code word length; K: message length; T: error-correction capability

Figure 1.4.: Properties BCH error correcting code.

the codeword changes on a small number of random bits, BCH decoding can still extract the original message. To encode a bit-string of for example 99 bits with an error correcting capability of 4 bits a codeword of 127 bits is needed. 4 bits in the 127 bits codeword can change and the 99 bit message can still be extracted. In practice this works as follows

- Secret  $\mathbf{S}$  is randomly generated and will be used as the shared secret
- $\mathbf{S}$  is encoded to a larger codeword using BCH encoding
- During enrollment  $\mathbf{W}_{\text{bch}}$  is created by XOR-ing  $\overline{\mathbf{X}}$  and the codeword, and stored in the database.
- During key extraction,  $\mathbf{M}$  is XOR-ed with  $\mathbf{W}_{\text{bch}}$ . Since  $\mathbf{M}$  and  $\overline{\mathbf{X}}$  should almost be the same, this leads to the codeword with a small number of bits changed. BCH decoding can be used to extract the shared secret if the number of bits changed is equal to or smaller than the error correcting capability.

A similar method can also be used without adding confusion (the secret  $\mathbf{S}$ ). In that case  $\mathbf{M}$  is used in BCH decoding to directly generate a smaller secret. The number of differences present between all  $\mathbf{X}$  during enrollment determine how big this secret can be. It is much better to find all unsteady bits during enrollment since each error correcting bit that is needed takes about (in this case) 7 bits from the message length. However, to obtain all unreliable bits during enrollment it should be possible to influence the environment variables (e.g. temperature should be varied) which might not be possible for all variables.

### 1.6.2. Quantized Key Extraction

By taking a contracting function it is possible to use all measurement values and not throw away any ‘unsteady’ points. The idea is that with the enrollment information, helper data is created that shifts measurement values first to lattice points [19](see figure 1.5). Within a

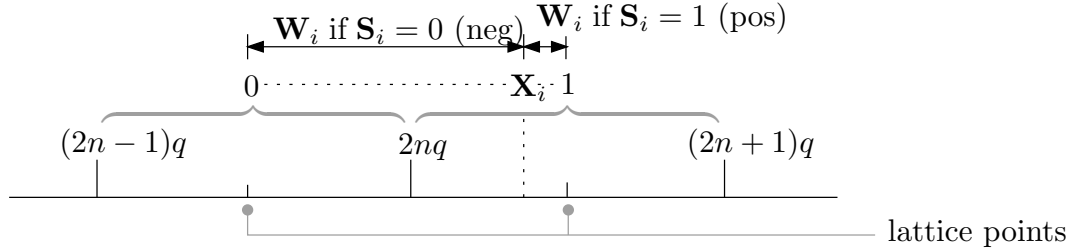


Figure 1.5.: Quantized key extraction:  $X_i$  is first shifted to a lattice point by adding  $W_i$ .

range around these lattice points, shifted values quantize to the same key value. Measurement values are less likely to be near the thresholds (unless the contraction is chosen too small). Consider the following  $\delta$ -contracting function:

$$\mathbf{W}_i = \begin{cases} (2n - \frac{1}{2})q - \mathbf{X}_i & \text{if } \mathbf{S}_i = 0 \\ (2n + \frac{1}{2})q - \mathbf{X}_i & \text{if } \mathbf{S}_i = 1 \end{cases} \quad n \text{ such that } -q \leq \mathbf{W}_i < q \quad (1.4)$$

$$\mathbf{S}_{meas,i} = \begin{cases} 0 & \text{if } (2n - 1)q \leq \mathbf{X}_i + \mathbf{W}_i < 2nq \\ 1 & \text{if } 2nq \leq \mathbf{X}_i + \mathbf{W}_i < (2n + 1)q \end{cases} \quad \text{for any } n = \dots, -1, 0, 1 \dots \quad (1.5)$$

with:

- $\mathbf{W}_i$  : Helper data (created during enrollment)
- $i$  :  $i$ -th dimension of vector
- $q$  : Quantization step size
- $\mathbf{X}_i$  : Measured value
- $\mathbf{S}_{meas,i}$  : Derived key value (recognition phase)

By using this contraction function,  $\mathbf{X}_i + \mathbf{W}_i$  is pushed to the nearest lattice point  $((2n + \frac{1}{2})q$  or  $(2n - \frac{1}{2})q$ ). And  $x_i + \mathbf{W}_i + \delta$  will be quantized to the same  $\mathbf{S}_i$  for small values of  $\delta$ . The contraction range  $\delta$  equals  $\pm q/2$ .

If the range of values of  $\mathbf{X}_i$ 's of different objects  $\sigma_X^2$ , is big compared to the noise  $\sigma_n^2$ , then it is beneficial to quantize to more than two values. The number of values that is quantized



to is called  $v$  ( $2 \leq v$ ). Equations 1.4 and 1.5 can be generalized to:

$$\mathbf{W}_i = \begin{cases} (vn + a_{S_i})q - \mathbf{X}_i & \text{if } \mathbf{S}_i = 0 \\ \dots & \dots \\ (vn + a_{S_i})q - \mathbf{X}_i & \text{if } \mathbf{S}_i = v - 1 \end{cases} \quad n \text{ such that } -\frac{v}{2}q \leq \mathbf{W}_i < \frac{v}{2}q \quad (1.6)$$

$$a_j = \begin{cases} -\frac{j}{2} - \frac{1}{2} & \text{if } j \bmod 2 = 0 \\ \frac{j}{2} & \text{if } j \bmod 2 = 1 \end{cases} \quad \text{for } 0 \leq j < v \quad (1.7)$$

$$\mathbf{S}_{meas,i} = j \quad \text{if } (vn + a_j - \frac{1}{2})q \leq \mathbf{X}_i + \mathbf{W}_i < (vn + a_j + \frac{1}{2})q \quad \text{for any } n = \dots, -1, 0, 1, \dots \quad (1.8)$$

with:

$v$  : Number of different values that is quantized to

$a_j$  : Addition to  $vn$  depending on the value of  $j$ . See table 1.3 for some values of  $a_j$  for different  $j$ .

$j$	$a_j$	$j$	$a_j$
0	-0.5	4	-2.5
1	0.5	5	2.5
2	-1.5	6	-3.5
3	1.5	7	3.5

Table 1.3.: Some values of  $a_j$  for different  $j$ .

The values for  $a_j$  can be chosen different too. A consequence of the choice for  $a_j$  as in equation 1.7, is that the lattice points that are next to each other do not quantize to values  $0, 1, \dots, v - 1, 0$ , but the order of the values that belong to the lattice points is different. See figure 1.6 for an example with  $v = 4$ . The measurement values that are divisible by  $4q$  are indicated by  $4n_0q$ ,  $4n_1q$  and  $4n_2q$ .

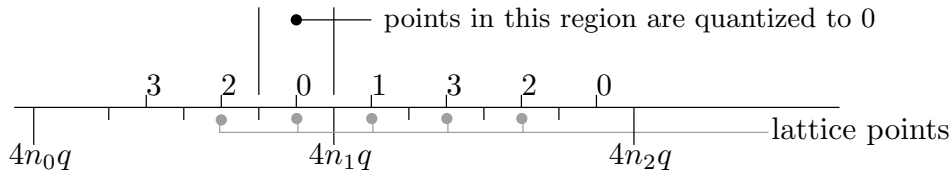


Figure 1.6.: Order of quantization values. Example for  $v = 4$ .

On a computer a different method with the same characteristics can be used, since this method is easier to compute. The method will be explained with by an example ( $v = 2$ ). Consider figure 1.7:

During enrollment the measured value  $\mathbf{X}_i$  is first quantized to  $\mathbf{Y}_i$ , which means all values in a range  $[2nq, (2n + 1)q)$  get the same integer value ( $2n$ ).  $\mathbf{Y}_i$  is then converted to a key value  $\mathbf{Z}_i (= \mathbf{Y}_i \bmod 2^k)$ . This key value can actually already be used, but in order to create

more random keys, the secret is chosen and the helper data is adjusted to produce the chosen secret from this data. Suppose in this example the chosen secret  $\mathbf{S}_i = 1$ , then  $\mathbf{Z}_i$  must be shifted with  $d2\mathbf{S}_i$ , which depends on the minimum of  $|d2\text{left}\mathbf{S}_i|$  and  $|d2\text{right}\mathbf{S}_i|$  (in this case  $d2\text{left}\mathbf{S}_i$  is chosen). This can be done for all  $i: 0 < i \leq l$  to generate the helper data. But the helper data should also shift  $\mathbf{X}_i$  first to a lattice point, so finally the helper data is:

$$d2\mathbf{S}_i = \begin{cases} d2\text{left}\mathbf{S}_i & \text{if } |d2\text{left}\mathbf{S}_i| \leq |d2\text{right}\mathbf{S}_i| \\ d2\text{right}\mathbf{S}_i & \text{if } |d2\text{left}\mathbf{S}_i| > |d2\text{right}\mathbf{S}_i| \end{cases} \quad (1.9)$$

$$\mathbf{W}_i = d2\text{lattice} + q \cdot d2\mathbf{S}_i \quad (1.10)$$

with:

- $\mathbf{W}_i$  : Derived helper data
- $d2\text{lattice}$  : Distance from  $\mathbf{X}_i$  to closest lattice point
- $q$  : Quantization step. Can actually be dependent on the measurement value, so  $q$  could be dependent on  $i$  as well and be replaced by  $\mathbf{Q}_i$ .
- $d2\mathbf{S}_i$  : Distance to closest  $\mathbf{S}_i \pmod{2^k}$

This helper data is used to produce  $\mathbf{S}_{meas,i}$  during authentication and/or key extraction.

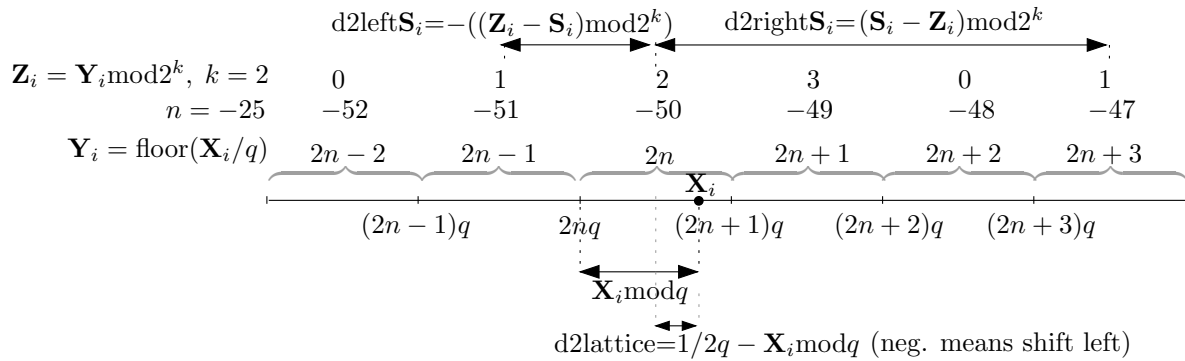


Figure 1.7.: Example of quantized key extraction.

### Probability of Error

When extracting keys for encryption/decryption it is of course very important that the key is always the same. Suppose a secret message  $s = \{0, 1\}$  is verified. The probability there is an error in one dimension equals:

$$P_e = 2Q\left(\frac{q}{2\sigma_n}\right) - 2Q\left(\frac{3q}{2\sigma_n}\right) + 2Q\left(\frac{5q}{2\sigma_n}\right) - \dots \quad (1.11)$$

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \quad (1.12)$$

with:

- $P_e$  : Probability of error in one dimension

- $q$  : Quantization step size  
 $\sigma_n$  : Standard deviation of the noise  
 $Q(x)$  : Integral over the Gaussian probability density function (unity variance)

When  $s$  can consist of a larger range of values  $v$ , this changes to:

$$P_e = 2Q\left(\frac{(0v + 1/2)q}{\sigma_n}\right) - 2Q\left(\frac{(1v - 1/2)q}{\sigma_n}\right) + 2Q\left(\frac{(1v + 1/2)q}{\sigma_n}\right) - \dots \quad (1.13)$$

Note that for  $v = 2$  this gives equation 1.11 on the preceding page. The chance for error becomes higher when  $v$  increases. Figure 1.8 shows a plot of both equations.

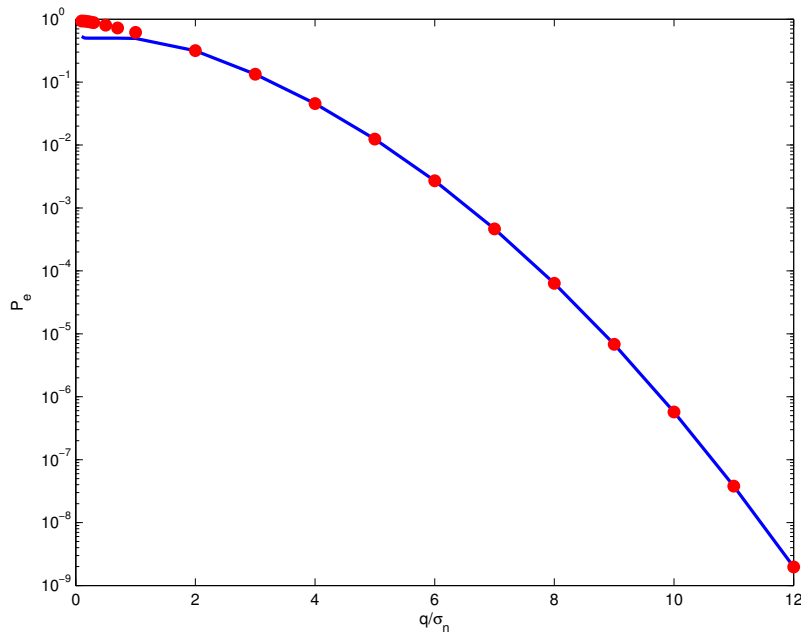


Figure 1.8.: Error probability per dimension as function of  $q/\sigma_n$ . solid line:  $v = 2$ , dotted:  $v = 16$ .

### Leakage of Information

In this section the leakage of information is calculated for the assumption that the input signal  $\mathbf{X}$  is normally distributed. The statistical behavior of  $\mathbf{W}$  is determined by those of  $\mathbf{X}$  and  $\mathbf{S}$ . For convenience ' $a$ ' is written instead of ' $\mathbf{A}_i$ ' for variables with a subscript  $i$  in them.

For  $s = 1$ ,  $x = (2n + 1/2)q - w$  (eq. 1.4), so:

$$f_W(w|s = 1) = \begin{cases} 0 & \text{for } |w| > q \\ \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{((2n+1/2)q-w)^2}{2\sigma_x^2}} & \text{for } |w| \leq q \end{cases} \quad (1.14)$$

$$f_W(w|s = 0) = \begin{cases} 0 & \text{for } |w| > q \\ \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{((2n-1/2)q-w)^2}{2\sigma_x^2}} & \text{for } |w| \leq q \end{cases} \quad (1.15)$$

$$f_W(w) = f_W(w|s = 1)P(s = 1) + f_W(w|s = 0)P(s = 0) \quad (1.16)$$

$$p_{w0} = P(s = 0|W = w) = \frac{f_W(w|s = 0)}{f_W(w)}P(s = 0) \quad (1.17)$$

with:

- $f_W(w|s = \dots)$  : Conditional probability density function (cpdf) of helper data value
- $f_W(w)$  : Probability density function (pdf)
- $\sigma_x$  : Standard deviation of inter-PUF variation
- $p_{w0}$  : A posteriori probability (using Bayes rule:  $P(R = r|e) = \frac{P(e|R=r)P(R=r)}{P(e)}$  with  $P(e) = \sum_r P(e|R = r) \cdot P(R = r)$ ),  $p_{w1}$  is defined similarly.

Figure 1.9 on the facing page plots  $q \cdot f_W(w|s = 0)$  and  $q \cdot f_W(w|s = 1)$  as function of  $w/q$  (see appendix B.1 for rewriting). Information leaks whenever  $f_W(w|s = 0) \neq f_W(w|s = 1)$ . One can see from the figure that the higher  $q/\sigma_x$ , the more information is leaked. From the figure it might seem that the unconditional probability function  $f_W(w)$  is constant, but it's not. Just as that  $f_W(w|s = 0) + f_W(-w|s = 0) = 1$  is not true (although for some  $w$  it is). From the formulas 1.14 to 1.17 on the current page the mutual information between  $w$  and  $s$  can be derived:

$$\begin{aligned} H(S) &= -\sum_i P(s = i) \log_2 P(s = i) \\ &= -2(0.5 \log_2 0.5) = 1 \end{aligned} \quad (1.18)$$

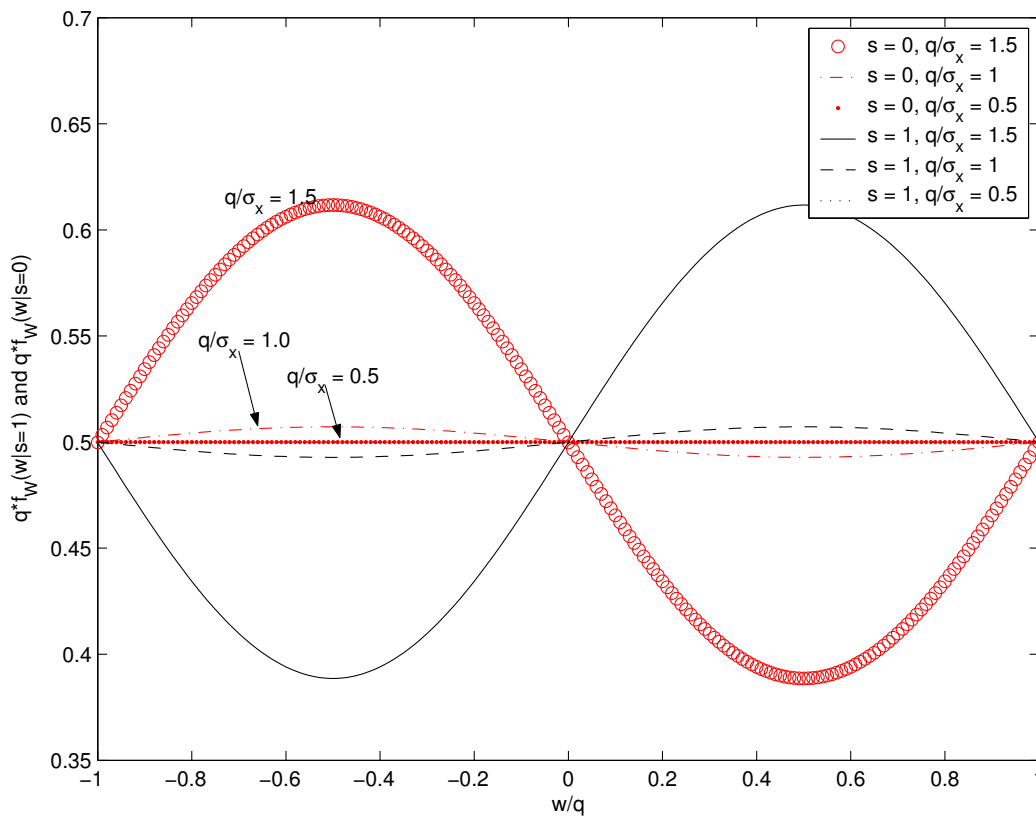


Figure 1.9.: Conditional probability density functions  $q \cdot f_W(w|s=0)$  and  $q \cdot f_W(w|s=1)$  as function of  $w/q$ . Functions plotted for different values of  $q/\sigma_x$ .

$$I(W; S) = H(S) - \int_{-q}^q H(S|W = w) f_W(w) dw \quad (1.19)$$

$$= 1 + \int_{-q}^q (p_{w1} \log_2 p_{w1} + p_{w0} \log_2 p_{w0}) f_W(w) dw \quad (1.20)$$

$$= 1 + \frac{1}{2} \int_{-q}^q f_W(w|s=0) \log_2 \left( \frac{f_W(w|s=0)}{f_W(w)} \cdot \frac{1}{2} \right) dw$$

$$+ \frac{1}{2} \int_{-q}^q f_W(w|s=1) \log_2 \left( \frac{f_W(w|s=1)}{f_W(w)} \cdot \frac{1}{2} \right) dw \quad (1.21)$$

$$= 1 + \frac{1}{2} \int_{-q}^q f_W(w|s=0) \log_2 f_W(w|s=0) dw$$

$$+ \frac{1}{2} \int_{-q}^q f_W(w|s=1) \log_2 f_W(w|s=1) dw$$

$$- \int_{-q}^q f_W(w) \log_2(2f_W(w)) dw \quad (1.22)$$

$$= \int_{-q}^q f_W(w|s=1) \log_2 f_W(w|s=1) dw - \int_{-q}^q f_W(w) \log_2 f_W(w) dw \quad (1.23)$$

with:

$H(S)$  : Information theoretic entropy of a discrete random variable  $S$  [bits]

$I(W; S)$  : Mutual information between  $W$  and  $S$

Figure 1.11 on page 24 shows the mutual information between  $W$  and  $S$  plotted against  $\sigma_x/q$  (see appendix B.2 for rewriting the equation). The lowest line in the figure ( $v = 2$ ), represents the value for the mutual information just calculated.

### Leakage of Information for Quantization to Multiple Values

The mutual information between  $W$  and  $S$  can be generalized when the measurement values are quantized to  $v$  values. The equations then change to:

$$f_W(w|s=j) = \begin{cases} 0 & \text{for } |w| > \frac{v}{2}q \\ \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{((vn+a_j)q-w)^2}{2\sigma_x^2}} & \text{for } |w| \leq \frac{v}{2}q \end{cases} \quad (1.24)$$

$$f_W(w) = \sum_{j=0}^{v-1} f_W(w|s=j)P(s=j) \quad (1.25)$$

$$p_{wj} = P(s=j|W=w) = \frac{f_W(w|s=j)}{f_W(w)}P(s=j) \quad (1.26)$$

$$\begin{aligned} H(S) &= -\sum_i P(s=i) \log_2 P(s=i) \\ &= -v \left( \frac{1}{v} \log_2 \frac{1}{v} \right) = \log_2 v \end{aligned} \quad (1.27)$$

$$\begin{aligned}
I(W; S) &= H(S) - \int_{-\frac{v}{2}q}^{\frac{v}{2}q} H(S|W = w) f_W(w) \, dw & (1.28) \\
&= H(S) + \int_{-\frac{v}{2}q}^{\frac{v}{2}q} \sum_{j=0}^{v-1} (p_{wj} \log_2 p_{wj}) f_W(w) \, dw \\
&= H(S) + \frac{1}{v} \sum_{j=0}^{v-1} \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w|s = j) \log_2 \left( \frac{f_W(w|s = j)}{f_W(w)} \cdot \frac{1}{v} \right) \, dw \\
&= H(S) + \frac{1}{v} \sum_{j=0}^{v-1} \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w|s = j) (\log_2 f_W(w|s = j) - \log_2 v - \log_2 f_W(w)) \, dw \\
&= H(S) + \frac{1}{v} \sum_{j=0}^{v-1} \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w|s = j) \log_2 f_W(w|s = j) \, dw \\
&\quad - \frac{1}{v} \sum_{j=0}^{v-1} \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w|s = j) \log_2 v \, dw - \frac{1}{v} \sum_{j=0}^{v-1} \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w|s = j) \log_2 f_W(w) \, dw \\
&= H(S) + \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w|s = 1) \log_2 f_W(w|s = 1) \, dw \\
&\quad - \log_2 v \int_{-\frac{v}{2}q}^{\frac{v}{2}q} \sum_{j=0}^{v-1} \left[ \frac{1}{v} \cdot f_W(w|s = j) \right] \, dw - \int_{-\frac{v}{2}q}^{\frac{v}{2}q} \sum_{j=0}^{v-1} \left[ \frac{1}{v} \cdot f_W(w|s = j) \right] \log_2 f_W(w) \, dw \\
&= \log_2 v + \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w|s = 1) \log_2 f_W(w|s = 1) \, dw \\
&\quad - \log_2 v \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w) \, dw - \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w) \log_2 f_W(w) \, dw \\
&= \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w|s = 1) \log_2 f_W(w|s = 1) \, dw - \int_{-\frac{v}{2}q}^{\frac{v}{2}q} f_W(w) \log_2 f_W(w) \, dw & (1.29)
\end{aligned}$$

Figure 1.10 plots all  $q \cdot f_W(w|s = j)$  for  $v = 4$  and  $q/\sigma_x = 0.5$ . The figure shows that for  $v = 4$  more information is leaked than for  $v = 2$  (see figure 1.9). This can be seen from the fact that the cpdf's differ more in value (and more difference means more leakage). For  $q/\sigma_x = 0.5$  the difference between the maximum and minimum value for  $v = 2$  is practically zero, while for  $v = 4$  this difference is approximately 0.0072.



The mutual information for different values of  $v$  is plotted against  $\sigma_x/q$  in figure 1.11. The

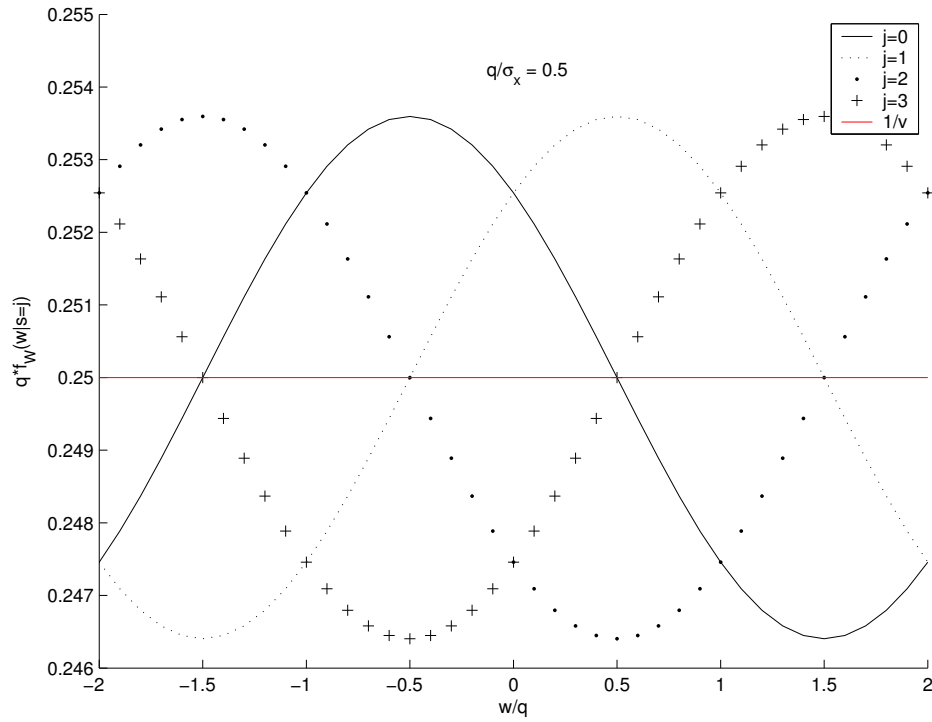


Figure 1.10.: Conditional probability density functions for  $v = 4$  as function of  $w/q$ .

figure also shows that for higher values of  $v$ , more information is leaked at the same  $\sigma_x/q$ .

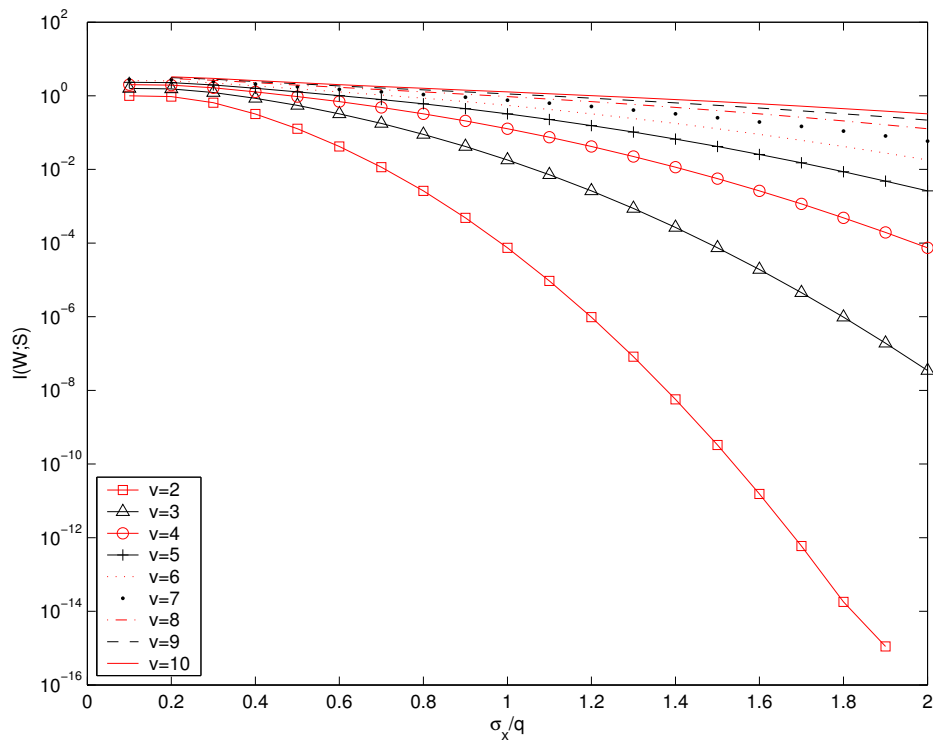


Figure 1.11.: Mutual information  $I(W;S)$  as function of  $\sigma/q$  for different values of  $v$ .

## 2. Glass Delay Line as APUF

For the experiments described in this report a glass delay was chosen to perform as an acoustical physical uncloneable function (APUF). This device was chosen, because it is an already existing device which converts electrical signals into (ultra)sound, transports the waves through a medium and converts them back again (which is exactly what an APUF should do). This section describes glass delay lines in general and how they can be used as a PUF.

### 2.1. Delay Lines in General

Delay lines are used to delay signals in for example television signals. In ultrasonic DLs the electrical signal (oscillating voltage) that has to be delayed is transformed to an identical mechanical vibration (described by the same time-function), which propagates as a wave through a liquid or solid medium. The name ‘ultrasonic’ is used because the frequencies involved are well above  $20kHz$  so the mechanical vibrations are in the ultrasonic area. The propagation speed of the mechanical vibration lies between 1 and  $6km/s$  which is about a factor  $10^5$  smaller than the propagation speed of electrical signals in coaxial cable. After the oscillation has undergone the desired delay, the vibration is transformed back to an electrical signal. Solid material (and liquid) DLs have some characteristics concerning wave propagation and transmission of the energy of the electrical signal to the transducer and back again [13].

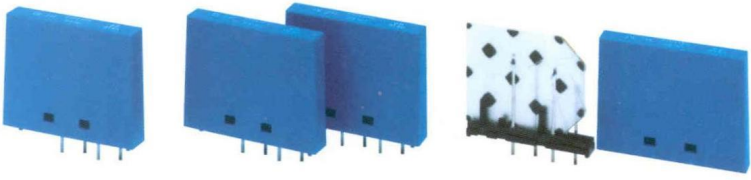
To transform the electrical signal to mechanical vibration and vice versa, transducers are used. Transducers are based on the piezo-electric effect. Characteristics of a DL depend greatly on the size and orientation of the transducers. Imagine a circular transducer of diameter  $d$  that emits ultrasound of wavelength  $\lambda$  in a boundless medium. If the transducer is a point source, it emits sound waves spherically and the transducer would receive the fraction of the energy that is related to the space angle under which the source sees the receiver. If the source is not a point source it can be shown that the energy is bundled parallel to the axis of the source until a distance of about  $d^2/2\lambda$ . This area is called the *Fresnel-zone* (near-field). When a receiving transducer with the same diameter  $d$  is placed within this zone, it will practically receive all energy. For larger distances the energy is distributed according to a certain function.

To obtain the wanted delay by a DL, it is not unusual that the path needs to be several decimeters or meters. It would be impossible to construct a DL with a straight path of this length, hence the path usually is folded up: the waves are reflected several times on the sides of the medium before they reach the receiving transducer.

Sometimes the main signal is not only reflected on the sides but also reflected at the receiving transducer. It then travels back to the input transducer and back to the output transducer again, generating a ‘third-time-round’ signal.

Figure 2.1 shows some characteristic information about several glass delay lines. One of these delay lines, the DL701 is used for the experiments.

### Glass delay lines



Type	DL 701 / DL 711	DL 750	DL 63	DL 720
Cat. number	4322 027 84772 / 84782	4322 027 84752	4322 027 8463	4322 027 84721
Product type	glass delay line	glass delay line	glass delay line	glass delay line
Case size	37 mm × 28 mm	37 mm × 28 mm	37 mm × 28 mm	37 mm × 28 mm
Construction	vertical 4 pins $\varnothing$ 0.8 mm × 4 mm in two offset rows. pin spacing 5.08 mm pin separation 2.54 mm lockfit pins available on request	vertical 4 pins $\varnothing$ 0.8 mm × 4 mm in two offset rows. pin spacing 5.08 mm pin separation 2.54 mm lockfit pins available on request	vertical 4 pins $\varnothing$ 0.8 mm × 4 mm in two offset rows. pin spacing 5.08 mm pin separation 2.54 mm lockfit pins available on request	vertical 4 pins $\varnothing$ 0.8 mm × 4 mm in two offset rows. pin spacing 5.08 mm pin separation 2.54 mm lockfit pins available on request
Applications	in colour TVs and monitors : chrominance delay (64 $\mu$ s) to compensate for phase distortion during transmission  in video recorders : drop out compensation circuitry	in colour TVs and VCRs for NTSC television  use in Comb Filter circuits	in colour TVs and monitors for the Brazilian TV system : chrominance delay for one line (64 $\mu$ s)	in colour TVs and monitors for the Argentinian TV system : chrominance delay for one line (64 $\mu$ s)
System	PAL / SECAM	NTSC	PAL M, Brazil	PAL N, Argentina
Delay time	63.943 $\pm$ 0.005 $\mu$ s	63.555 $\pm$ 0.005 $\mu$ s	63.486 $\pm$ 0.005 $\mu$ s	63.929 $\pm$ 0.005 $\mu$ s
Centre frequency	4.433619 MHz	3.579545 MHz	3.575611 MHz	3.582056 MHz
Insertion loss	9 $\pm$ 3 dB	9 $\pm$ 3 dB	9 $\pm$ 3 dB	9 $\pm$ 3 dB
Operating temperature range	- 20 to + 70 °C	- 20 to + 70 °C	- 20 to + 70 °C	- 20 to + 70 °C

Figure 2.1.: Information about glass delay lines [10].

## 2.2. The DL701

The DL701 is a delay line made by Philips that used to be used as a delay line in televisions. In figure 2.1 on the preceding page its main characteristics are listed and figure 2.2 shows an opened DL701. The signal in the DL701 is folded up in the device to create a longer delay time with a small device. The working of the device is as follows (see also figure 2.3):

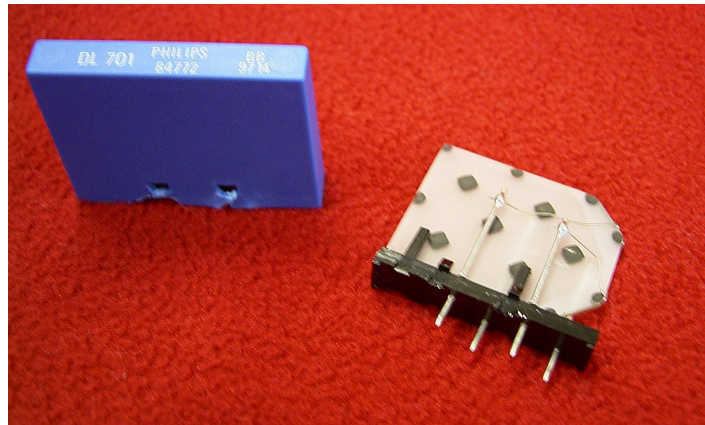


Figure 2.2.: The DL701.

- Electrical waves are sent to the input transducer of the DL701
- transducer converts electrical waves to mechanical waves
- mechanical waves travel through the medium (mainly as shear waves)
- the mechanical waves are reflected several times by the boundaries of the medium
- the mechanical waves are received by the output transducer
- transducer converts the mechanical waves back to electrical waves.

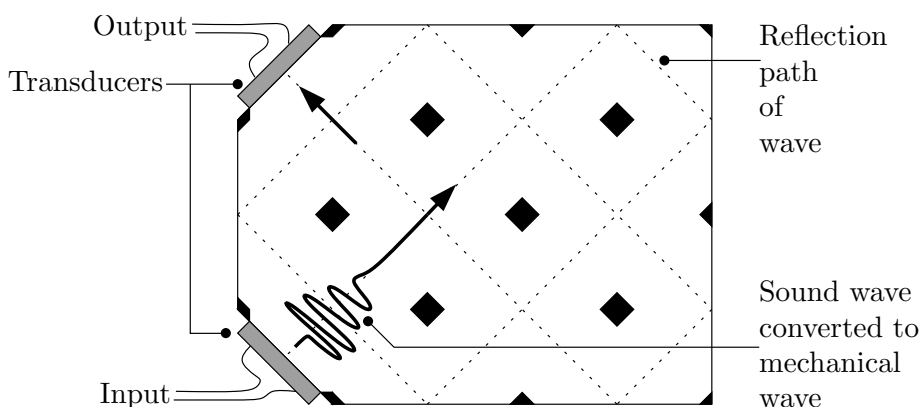


Figure 2.3.: Schematic view of the DL701

During the travel of the wave through the medium, the wave gets distorted by inhomogeneities in the medium (and by the conversion between electrical/mechanical/electrical wave). The

first goal of our experiments is to determine whether these inhomogeneities are unique enough to identify different DL701s. All the DL701s are made to be the same, but manufacturing randomness could be large enough to make this possible. In the end our goal is to investigate whether acoustical PUFs can be realized. The DL701 will function as a proof of concept.

## 2.3. Sound Propagation Through Solids

In solids, sound waves can propagate in four principle modes that are based on the way the particles oscillate. Sound can propagate as longitudinal waves, shear waves, surface waves, and in thin materials as plate waves. Longitudinal and shear waves are the two modes of propagation most widely used in ultrasonic testing. The particle movement responsible for the propagation of longitudinal and shear waves is illustrated in figure 2.4. Some properties of

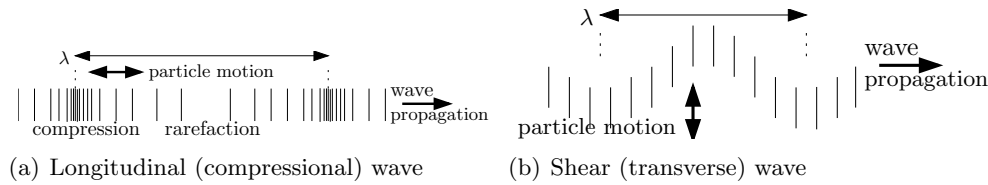


Figure 2.4.: Types of waves.

waves are wavelength, frequency, and velocity. Their relationship is expressed in the following equation:

$$\lambda = \frac{v}{f} \quad (2.1)$$

$$v = \sqrt{\frac{C_{ij}}{\rho}} \quad (2.2)$$

with:

$\lambda$  : wavelength

$v$  : velocity

$f$  : frequency

$\rho$  : material density

$C_{ij}$  : elastic constant. The subscript  $ij$  indicates the directionality of the elastic constants with respect to the wave type and direction of wave travel, because in anisotropic materials the elastic constants differ with each direction.

The wavelength of the ultrasound used has significant affect on the probability of detecting a discontinuity. A rule of thumb in industrial inspections is that discontinuities that are larger than one-half the size of wavelength can be usually be detected [7]. As frequency increases, sound tends to scatter from large or course grain structure and from small imperfections within a material.

Since more things in a material are likely to scatter a portion of the sound energy at higher frequencies, the penetrating power (or the maximum depth in a material that flaws can be located) is also reduced. Frequency also has an effect on the shape of the ultrasonic beam.

If comparing compressional and shear velocities it can be noted that shear velocity is approximately one half that of compressional. In delay lines almost exclusively shear waves are used [28]. Examples of velocities in different sorts of glass are shown in table 2.2.

<b>Name</b>	<b>Longitudinal Velocity</b> [ $cm/\mu s$ ]	<b>Shear Velocity</b> [ $cm/\mu s$ ]	<b>Density</b> [ $g/cm^3$ ]	<b>Acoustic Impedance</b> [ $g/cm^2 \cdot sec$ ]
Glass Crown (reg.)	.566	.352	2.60	14.5
Glass Crown (heaviest)	.526	.326	N/A	N/A
Glass Quartz	.557	.343	2.60	14.5
Glass Window	.679	.343	N/A	N/A
Glass, Plate	.571	N/A	2.75	10.7
Glass, Pyrex	.556	.198	N/A	N/A

Table 2.2.: Examples of velocities in materials.

## 2.4. DL701 as APUF

The DL701 can be probed with one certain frequency and the medium the wave travels through, changes the wave characteristic. Hopefully each DL701 produces a unique attenuation and/or phase change compared to other DL701s, as that would make each of them uniquely identifiable and more important: the response unpredictable. This uniqueness can for example be caused by inhomogeneities in the medium (and other variations e.g. in the transducers). Yet it is very unlikely that probing with one particular frequency will be enough to identify a large amount of DL701s, since the spread in response values has a limit which is related to this particular type of delay line. To uniquely identify one DL701 probably more challenges are needed at different frequencies. A collection of frequencies which can be used to identify a large amount of DL701s could be used as a challenge when a DL701 needs to be identified. Note that this challenge consists of multiple ‘mini-challenges’ at one frequency. Throughout this report the word ‘challenge’ refers to such a collection of mini-challenges and ‘response’ to its corresponding response. How many unique challenges one DL701 contains has to be investigated. Maybe the whole frequency-range of operation is needed to get a challenge that is unique enough for identification, it is unknown for now. When the DL701 has a very large challenge-response space and it is impossible to determine chosen CRPs from a polynomial number of plausible physical measurements, by an attacker who no longer has the device and that can only use a polynomial amount of resources, then the DL701 can be considered to be an APUF.

## 2.5. Number of Possible Structures

When the PUF is probed with (ultra)sound of wavelength  $\lambda$  it follows from the theory of waves in solids that only details of size  $\frac{1}{2}\lambda$  can be resolved (see section 2.3). The volume  $Ad$  of the PUF can be divided into elements of volume  $\lambda^3$  (voxels). In case of surface waves only the area of the PUF is used, which can be divided into elements of area  $\lambda^2$ . The following formulas can be derived [27]

$$N_{vox.3D} = Ad / \left(\frac{1}{2}\lambda\right)^3 \quad (2.3)$$

$$N_{vox.2D} = A / \left(\frac{1}{2}\lambda\right)^2 \quad (2.4)$$

$$\kappa = \{0, 1\}^{N_{vox}} \quad (2.5)$$

$$H(K) = N_{vox}h(f) \quad (2.6)$$

$$h(f) = -f \log f - (1 - f) \log(1 - f), h(f) \in [0, 1] \quad (2.7)$$

with:

$N_{vox.xx}$  : The number of voxels.  $xx = 3D$  for longitudinal waves,  $xx = 2D$  for surface waves.

$A$  : Area of the medium.

$\kappa$  : Bit-string that represents a PUF. It is assumed that the sound waves can only distinguish whether a voxel contains a scatterer or not. Internal degrees of freedom within a voxel can not be distinguished. This means a PUF (or equivalently  $\kappa$ ) can be represented as a bit-string of length  $N_{vox}$ .

$f$  : The fraction of the total volume that is filled with scattering material.

$h(f)$  : The binary entropy function of the filling fraction.

The information content of one voxel is at most one bit, which means a PUF is completely characterized if one knows which voxels contain a scatterer.

Since in delay lines shear waves are used, equation 2.4 can be used to calculate the number of unique structures that (theoretically) can be made with the DL701. It is not known exactly what the speed of sound is in this delay line, it is estimated as the path length the sound travels divided by the specified delay time.

$$\lambda = \frac{\text{pathlength}}{\text{delay} \cdot f} \quad (2.8)$$

$$\begin{aligned} N_{vox.2D} &= \frac{A}{\left(\frac{1}{2} \frac{\text{pathlength}}{\text{delay} \cdot f}\right)^2} \quad (2.9) \\ &= \frac{4A(\text{delay})^2 f^2}{(\text{pathlength})^2} \end{aligned}$$

For DL701:  $A = (24 * 30) - (6 * 6) = 684[mm^2]$ ,  $\text{pathlength} = 0.165[m]$ ,  $\text{delay} = 63.943[\mu s]$ , this gives  $N_{vox.2D} \approx 4.109 \cdot 10^{-10} \cdot f^2$ .

Which gives for several frequencies



---

Frequency [Mhz]	$N_{vox\_2D}$ [ $\cdot 10^3$ ]
2.0	1.6
4.4	8.0
8.2	28

Note that this means that for the DL701 there are about  $2^{28 \cdot 10^3}$  possible unique structures of the material (as seen by the system) *if* all random structures would appear during the manufacturing process (which of course is probably not the case) and more important *if* the system can distinguish all these structures with only 2 transducers. An attacker could use these  $28 \cdot 10^3$  bits to computationally simulate the output instead of storing all possible CRPs in advance. The whole system includes the transducers, which may show variations as well. Compared to the number of possible structures for an Optical PUF the number of structures is quite small. An optical PUF of  $1\text{cm}^3$  probed by light of a wavelength on the order of  $1\mu\text{m}$  has its structure specified with  $10^{-2}/10^{-6} = 10^{12}$  bits [23]. Even when an attacker knows all these bits she still has to be able to simulate the scattering for the waves. For light, simulating the scattering from one particle that is several times the wavelength, presently requires a supercomputer. For sound there are already programs that simulate the propagation of ultrasonic waves through multilayered structures. The difficulty of doing such for a delay line has not been investigated during this internship.



## 3. System Setup

This section describes the experimental setup that was created to perform measurements on APUFs. The system is called AKI — **A**PUF **K**ey extraction and **I**dentification system, and the main (Matlab) program is called AKI as well. Refer to the manual in section C to read more on the system and how to operate AKI.

### 3.1. Setup

Figure 3.1 shows the experimental setup that was built to perform measurements on the DL701 glass delay line. The system consists of several components which are schematically shown in figure 3.2. The schematic view shows that the communication between the computer

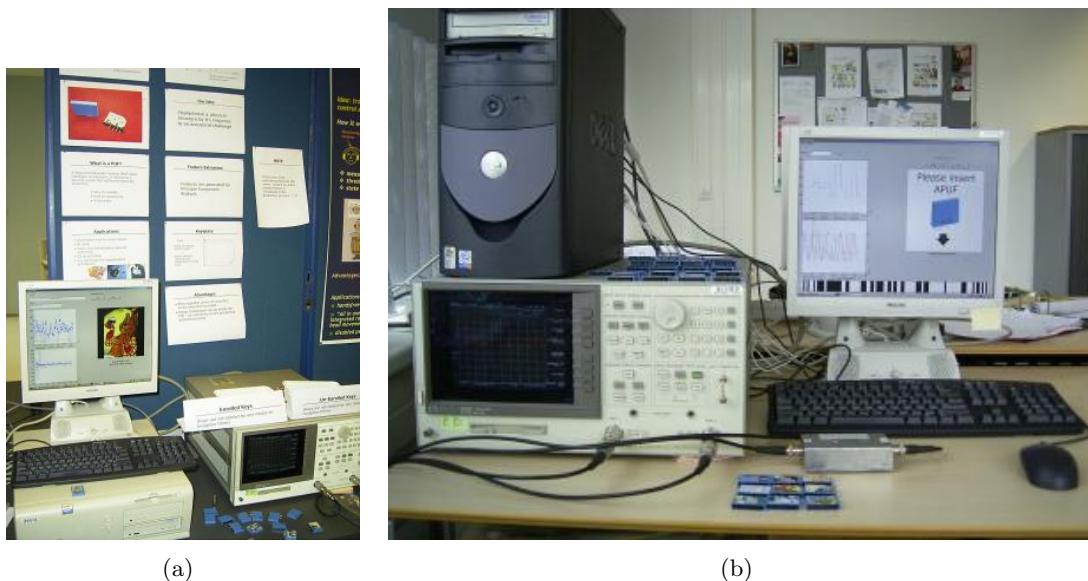


Figure 3.1.: The experimental setup. The left figure shows the system as it was used during a demonstration at Philips, the right as it was used during research. The rectangular device with the dark display is a network analyzer and the right figure also shows the reader to insert DL701s (the small iron-colored block in the front).

and network analyzer is done through a GPIB (General Purpose Interface Bus, IEEE 488) interface. This is a standard interface for communication between instruments from different sources. Appendix C.1 describes how to install the PCI card. So how is this system used? Below a rough description of its use to get an idea.

#### **Enrollment Phase (to learn the characteristics of an APUF)**

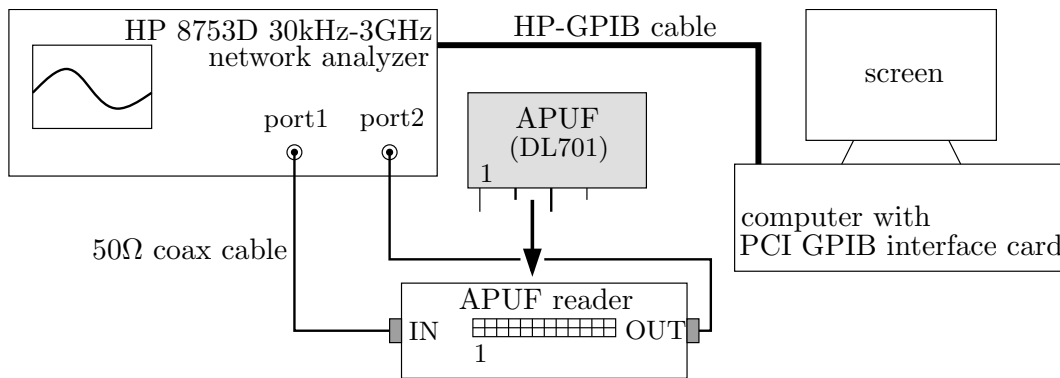


Figure 3.2.: Schematic overview of the system.

- Some (random) challenge is produced.
- The PUFs are one by one inserted and challenged several times with the challenge, the responses are saved.
- From this data, feature values or keys are extracted for each APUF.

#### Recognition (to identify an APUF)

- An APUF is inserted into the reader.
- The APUF is automatically challenged with a challenge that exists in the database and the response is recorded.
- The response is compared to the responses in the database. The APUF is identified as the APUF in the database with the closest (or identical) match. Note that above a certain threshold the APUF remains unidentified (this to prevent wrong identification of an APUF that is not in the database yet).

#### Key Extraction

- An APUF is inserted into the reader.
- The user selects the ID of the APUF from a list and tells the system to challenge it.
- The APUF is challenged and the response is recorded. With some helper data belonging to the given ID the response is transformed into a key which can for example be used as an encryption/decryption key.

More information about the specific steps that have to be made for enrollment and how to operate the system can be found in the AKI manual (see section C).

## 3.2. Communication

To communicate with the Network Analyzer, AKI calls the executable **Measure.exe**. Results from the analyzer are saved in a specified file and read in by AKI. **Enroll.exe** is used as a tool for enrollment for auto identification and threshold key extraction. The enrollment for

quantized key extraction is built into AKI. Both these executables were developed with the Borland 5.5 Compiler (BCC5.5) and the source code is included with the AKI program.

Measure.exe can take some command line options. These can be shown by entering: 'Measure.exe -h' in a command prompt window. Table 3.1 lists the options.

Usage of the options: Measure <option1> argument1 <option2>... The short option is always preceded by '-' a long option by '--'.

Option		Description
Short	Long	
0	close	take board offline
c	calibrate	calibration mode
f	find	find the gpib address of the connected device
h	help	this list
l	load	load limits, <i>-l 0</i> to clear limits
m	measure <int>	sweep & pass <int> times and get data trace
o	output <optional filename>	filename to write output data to
p	freq <optional filename>	freq. list to measure and sweep
s	sweep <optional filename>	load list to sweep

Table 3.1.: Command line options of Measure.exe.

Both executables are small and should not be difficult to understand if looked at the C-Code which contains comments. To understand and get an idea of how AKI operates, figures 3.3 and 3.4 show flow diagrams of the three main functionalities of AKI: they show what happens when the auto identification button, the threshold key extraction button or the quantized key extraction button is clicked.

For more information about the AKI system and how to operate AKI, please read the manual in section C.

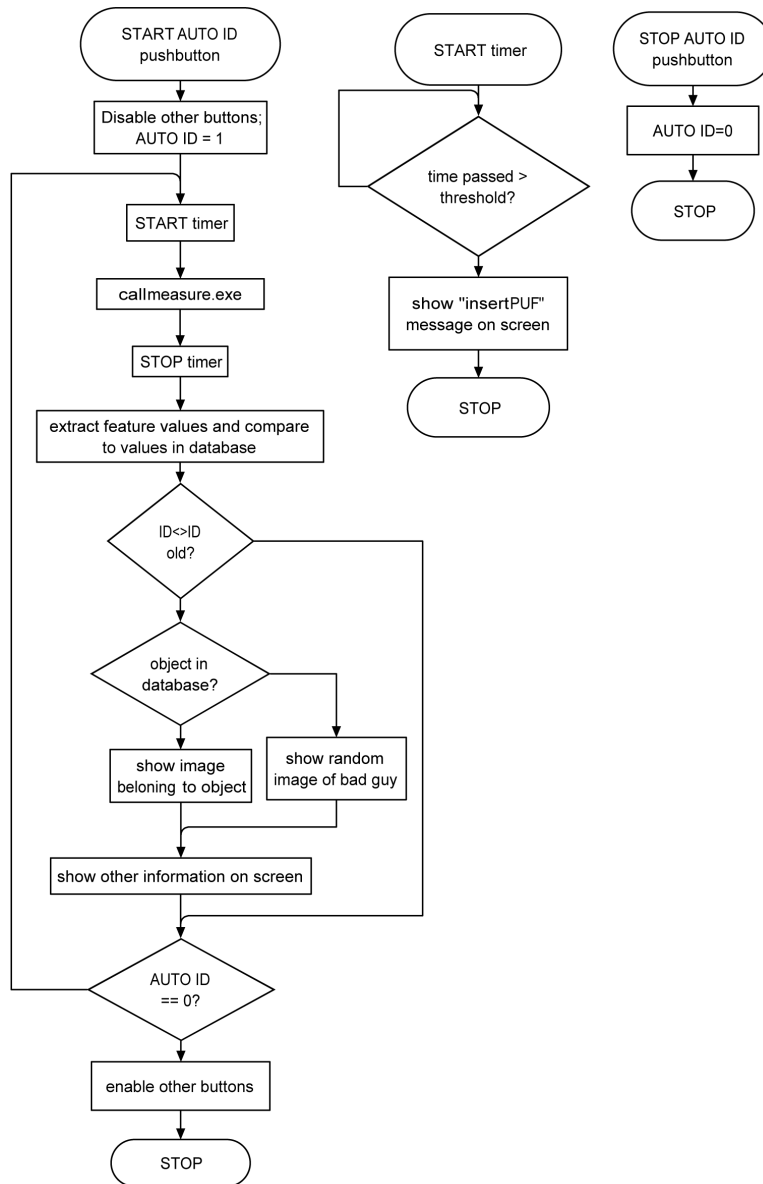


Figure 3.3.: Flow diagram of auto identification part of AKI.

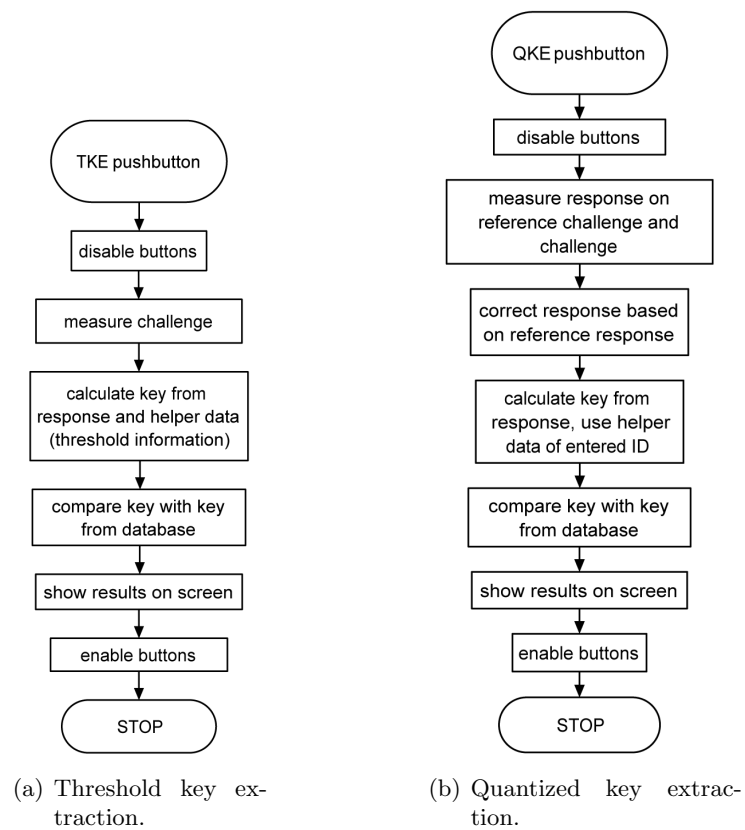


Figure 3.4.: Flow diagram of key extraction.





## 4. Experiments and Results

This section describes some results that were obtained with the AKI3 demo system. The demo system was used during the internal Philips Eureka Fair and during the Philips Corporate Research Exhibition to show the feasibility of APUFs. This section does not describe how to operate the system in order to obtain these results, for operation instructions refer to the manual (section C). The experiments are based on the three modes of operation AKI has

1. Identifying DL701s
2. Key extraction based on threshold
3. Key extraction based on quantization and shifting measurement values

One section is added before these to describe some typical measurements.

### 4.1. Typical Measurement

So what does a typical measurement of a DL701 look like? Figure 4.1 shows some images obtained from the network analyzer. The analyzer operates from  $30kHz$  to  $3GHz$ , but the useful frequency range is from about  $2.0MHz$  to  $8.2MHz$ . Outside these frequencies, a lot of noise is present in the signal. This can be seen from figure 4.1(d). The experiments will only be conducted within the  $2.0 - 8.2MHz$  frequency range.

#### 4.1.1. Characteristic Frequency Spectrum

By reducing the frequency range that is swept by the analyzer and keeping the number of measurement points the same, the frequency spectrum is 'zoomed in'. If that is done, a typical trend of the frequency spectrum can be seen as shown in figure 4.2(a). It looks like the frequency spectrum from the delay line has a characteristic sinus superimposed on another signal. The distance between two local maxima in the spectrum is about  $8.0 \pm 0.3kHz$  at  $4.4MHz$ . This characteristic reminds of a 'third-time-round' signal (see section 2.1 on page 25): the main signal is reflected at the output transducer, travels back to the input transducer and back to the output transducer again. This can be explained with a simple model as shown in figure 4.2(b)

$$H = \frac{e^{j\omega t} + e^{j\omega(t+\tau)}}{e^{j\omega t}} \quad (4.1)$$

$$= 1 + e^{j\omega\tau} \quad (4.2)$$

$$= (1 + \cos \omega\tau) + j \sin \omega\tau \quad (4.3)$$

$$|H| = \sqrt{(1 + \cos \omega\tau)^2 + \sin^2 \omega\tau} \quad (4.4)$$

$$= \sqrt{2 + 2 \cos \omega\tau} \quad (4.5)$$

with:

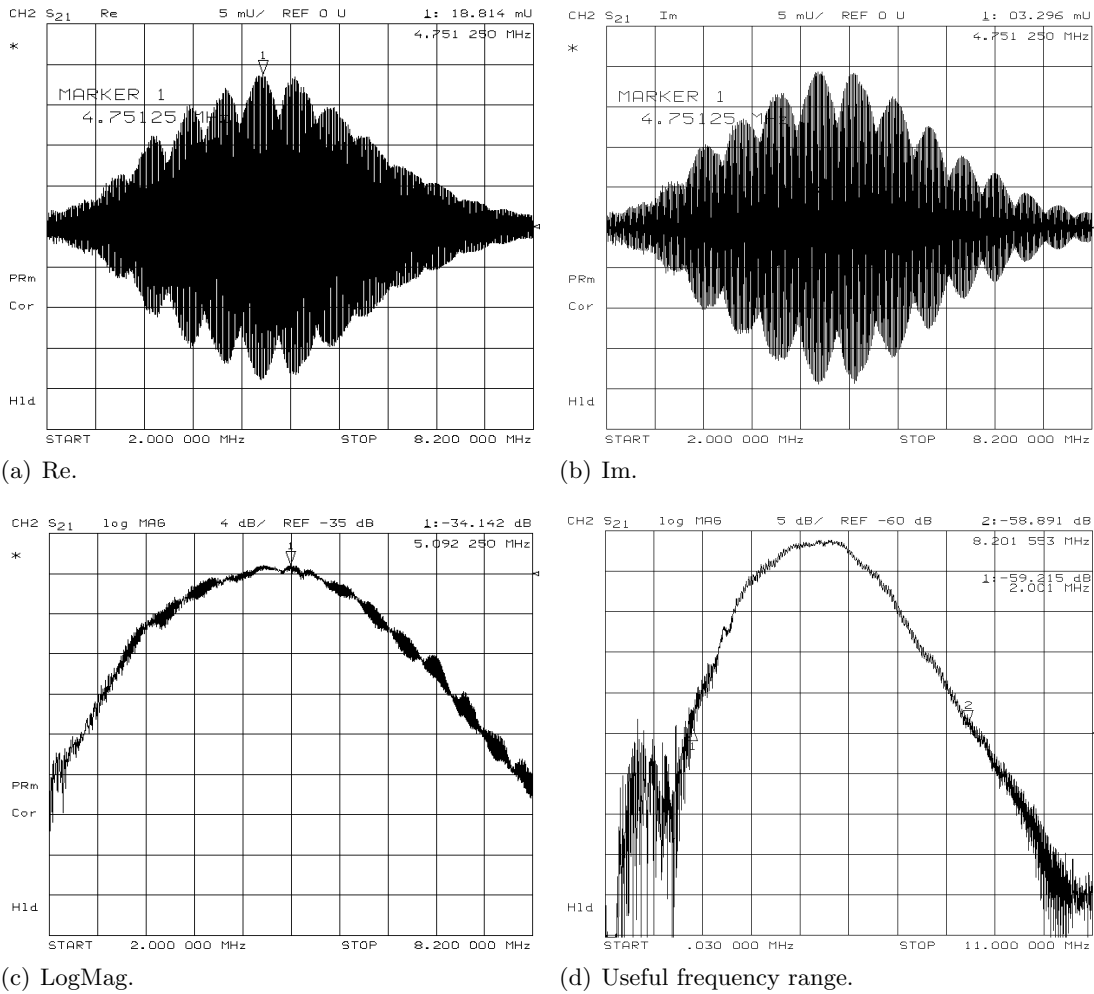


Figure 4.1.: Typical measurement values from the DL701.

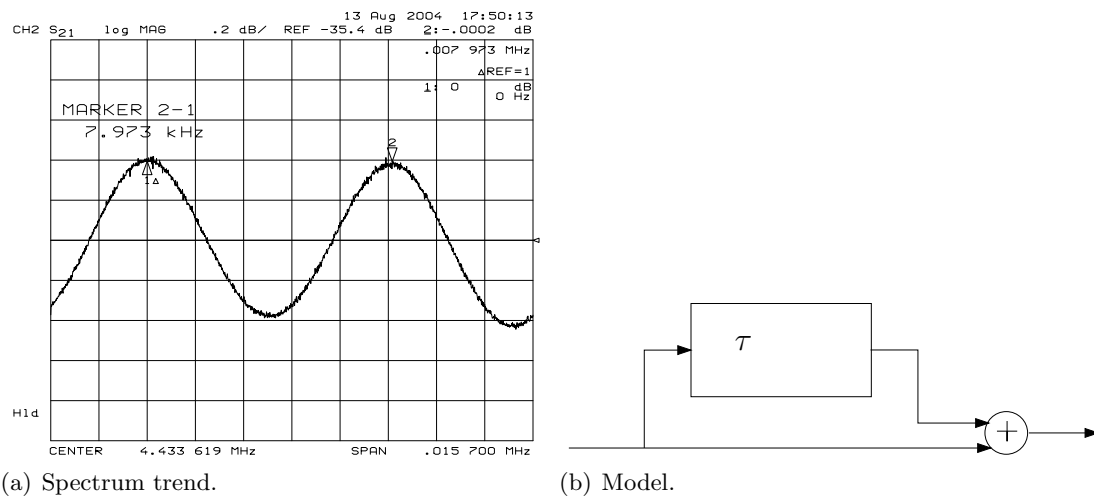


Figure 4.2.: Characteristic frequency spectrum.

- $|H|$  : Transfer function of amplitude  
 $e^{j\omega t}$  : The original signal  
 $\tau$  : Delay of the ‘third-time-round’ signal  
 $\omega$  :  $2\pi f$  (angular speed), with frequency  $f$

This transfer function is the same again after  $\omega\tau = 2\pi f\tau = 2\pi$ . In this case  $\tau$  is two times the delay time specified for the DL701. So  $f = 1/\tau = 1/(2 \cdot (64.943 \pm 0.005))[\mu s] = 7.8195 \pm 0.0007[kHz]$ . (64.943[ $\mu s$ ] taken from figure 2.1 on page 26). Which is in agreement with the measured signal and accounts for the characteristic sinus in the frequency spectrum.

## 4.2. Identification of DL701s

This section investigates whether the DL701s can be uniquely identified. In order to recognize the APUFs it is first necessary to determine certain characteristic features of each device. This is done by enrollment measurements, performing *Principle Component Analysis* (PCA, [9]) on the measurement results and saving the results. After this enrollment, identification can start. The following equations summarize what is done.

$$\mathbf{f} = (f_1, \dots, f_c) \quad (4.6)$$

$$\mathbf{g}_o = (g_{o1}, \dots, g_{oc}) \quad (4.7)$$

$$D_o = \frac{1}{c} \sum_{i=1}^c \frac{|f_i - g_{oi}|}{\sigma_i} \quad (4.8)$$

with:

- $\mathbf{f}$  : Feature values that are determined from a measurement.  $c$  indicates the number of features (components) used for identification.
- $\mathbf{g}_o$  : Feature values of object  $o$  (stored in database). These values were calculated from enrollment measurements.
- $D_o$  : Difference measure (distance) between  $\mathbf{f}$  and  $\mathbf{g}_o$ . If the difference is below a certain (experimentally determined) level, the measured object is identified as object  $o$ .
- $\sigma_i$  : Standard deviation of feature  $i$  (stored in database)

### Principle Component Analysis

Principal component analysis (PCA) is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components.

**Definition 4** *Principle Components* - A set of variables that define a projection that encapsulates the maximum amount of variation in a data set and is orthogonal (and therefore uncorrelated) to the previous principle component of the same data set [9].

The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. PCA has two main objectives

- To reduce and discover the dimensionality of the data set
- To identify new meaningful underlying variables

To explain this, look at the relation between points and their axis. Figure 4.3 shows some fictitious 2D data. The idea is to turn the axis in such a way, that the points form an ellipsoid from which the longest and shortest side overlap the new position of the axis. The axis where the values differ the most is the 1st principle component, the axis with the next most variability in the values the 2nd and so on (in the case of multi-dimensional data). For this

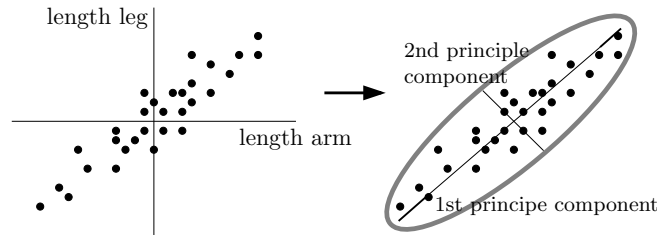


Figure 4.3.: PCA.

particular example the length of a leg and arm of a person are probably correlated, since on average, people with longer legs will also have longer arms. As a matter of fact it might be found that the first principle component represents the underlying variable ‘length’. From a math point of view, PCA is performed as follows

$$\mathbf{x}_i = (x_{i1}, \dots, x_{im})^T \quad 1 \leq i \leq c \quad (4.9)$$

$$\mathbf{o}_j = (x_{1j}, \dots, x_{cj}) \quad 1 \leq j \leq m \quad (4.10)$$

$$\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_c) \quad (4.11)$$

$$= (\mathbf{o}_1, \dots, \mathbf{o}_m)^T \quad (4.12)$$

$$\mu_{\mathbf{x}_i} = \frac{1}{m} \sum_{j=1}^m x_{ij} \quad (4.13)$$

$$c_{ij} = \frac{1}{m-1} (\mathbf{x}_i - \mu_{\mathbf{x}_i})^T (\mathbf{x}_j - \mu_{\mathbf{x}_j}) \quad 1 \leq i \leq j \leq c \quad (4.14)$$

with:

- $\mathbf{x}_i$  : Vector of several measurements of variable i (column vector)
- $\mathbf{o}_j$  : Vector of measurement values of observation j
- $\mathbf{X}$  : Matrix with  $\mathbf{o}_j$ 's as row vectors and  $\mathbf{x}_i$ 's as column vectors
- $c_{ij}$  : The covariance between the variable components  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , with  $c_{ii}$  the variance of component  $\mathbf{x}_i$ . If two components are uncorrelated their covariance is zero.  $\mathbf{C}_X$  is the covariance matrix of  $\mathbf{X}$ , which has  $c_{ij}$  as matrix values. The covariance matrix is always symmetric
- $\mu_{\mathbf{x}_i}$  : Mean of  $\mathbf{x}_i$
- $\dots^T$  : Transpose of ...

Now consider the matrix  $\mathbf{X}$  where each observation vector is a measurement of frequency points on one delay line. The full possible range of delay lines is called the population, one

measurement on one delay line is called a sample. From samples  $\mathbf{o}_1, \dots, \mathbf{o}_m$  the sample mean and the covariance matrix can be calculated as estimates of the mean and the covariance matrix. After that an orthogonal basis is calculated for the covariance matrix. For a symmetric matrix an orthogonal basis can be found by calculating its eigenvalues and eigenvectors. These can be found by solving

$$\mathbf{C}_X \mathbf{e}_i = \lambda_i \mathbf{e}_i \quad \text{for } i = 1, \dots, m \quad (4.15)$$

$$|\mathbf{C}_X - \lambda \mathbf{I}| = 0 \quad (4.16)$$

with:

- $\mathbf{e}_i$  : Eigenvectors of  $\mathbf{C}_X$  (column vectors)
- $\lambda_i$  : Eigenvalues of  $\mathbf{C}_X$
- $\mathbf{I}$  : Identity matrix with the same order (size) as  $\mathbf{C}_X$
- $|\dots|$  : The determinant of ...

The second equation (known as the characteristic equation of  $\mathbf{C}_X$ ) finds the solutions to the first. If the sample vector has  $c$  components,  $\mathbf{C}_X$  is of order  $c$  and the second equation also becomes of order  $c$ . By ordering the eigenvectors in the order of descending eigenvalues, an ordered orthogonal basis can be created, with the first eigenvector having the direction of largest variance of the data. In this way, the directions in which the data set has the most significant amounts of energy can be found.

Now assume a data set for which the sample mean and covariance matrix have been calculated. Let  $A$  be the matrix consisting of eigenvectors of the covariance matrix as the row vectors. It is possible to transform points from one coordinate system to the other:

$$\mathbf{f} = \mathbf{A}(\mathbf{o}^T - \mu_o) \quad (4.17)$$

$$\mathbf{o} = \mathbf{A}^{-1}\mathbf{f} + \mu_o \quad (4.18)$$

with:

- $\mathbf{f}$  : Point in the orthogonal coordinate system defined by the eigenvectors. Components of  $\mathbf{f}$  can be seen as the coordinates in the orthogonal base and are the feature values of the DL inserted during measurement of  $\mathbf{o}$ .
- $\mathbf{o}$  : Point in original coordinate system
- $A^{-1}$  : Inverse of  $A$ . Note that  $A^{-1} = A^T$ , because  $A$  is an orthogonal matrix.

Data may be represented in terms of only a few eigenvectors of the covariance matrix instead of using all the basis vectors of the orthogonal basis. This minimizes the mean-square error between the data and its representation for a given number of eigenvectors.

In order to calculate the feature values of a sample  $\mathbf{o}$ , helper data  $\mathbf{A}$  and  $\mu_o$  are needed. These are created from the enrollment measurements (just like  $\mathbf{g}$ ).

#### 4.2.1. Entropy of the Principle Components

Figure 4.4 shows the feature values of some principle components. These were determined by sweeping 65 DLs over a frequency range of 2.0–8.2MHz and performing PCA on the results. Each DL was measured ten times and the x-axis shows all subsequent measurements, so the

first ten measurements belong to DL00, the next ten to DL01 etc. Note that the set of ten measurements is divided into a train and test set. The train set is used for the PCA, the test set to check whether new measurements will be close to the determined feature values. In the plots, the first seven (blue) points belonging to one object are from the train set, the three last (red) points are from the test set. Figure 4.4(d) shows a close up of the feature value of one delay line. From the images it is seen that the feature values for each principle component differ much more between the objects than the values differ between each measurement of the same object (due to noise). Further, the absolute feature values decrease for higher principle components, but the noise also decreases. Principle component 65 does not help any more to distinguish delay lines, which is in agreement with the PCA method, since only 65 objects were used. In order to determine how much information is contained in each principle component,

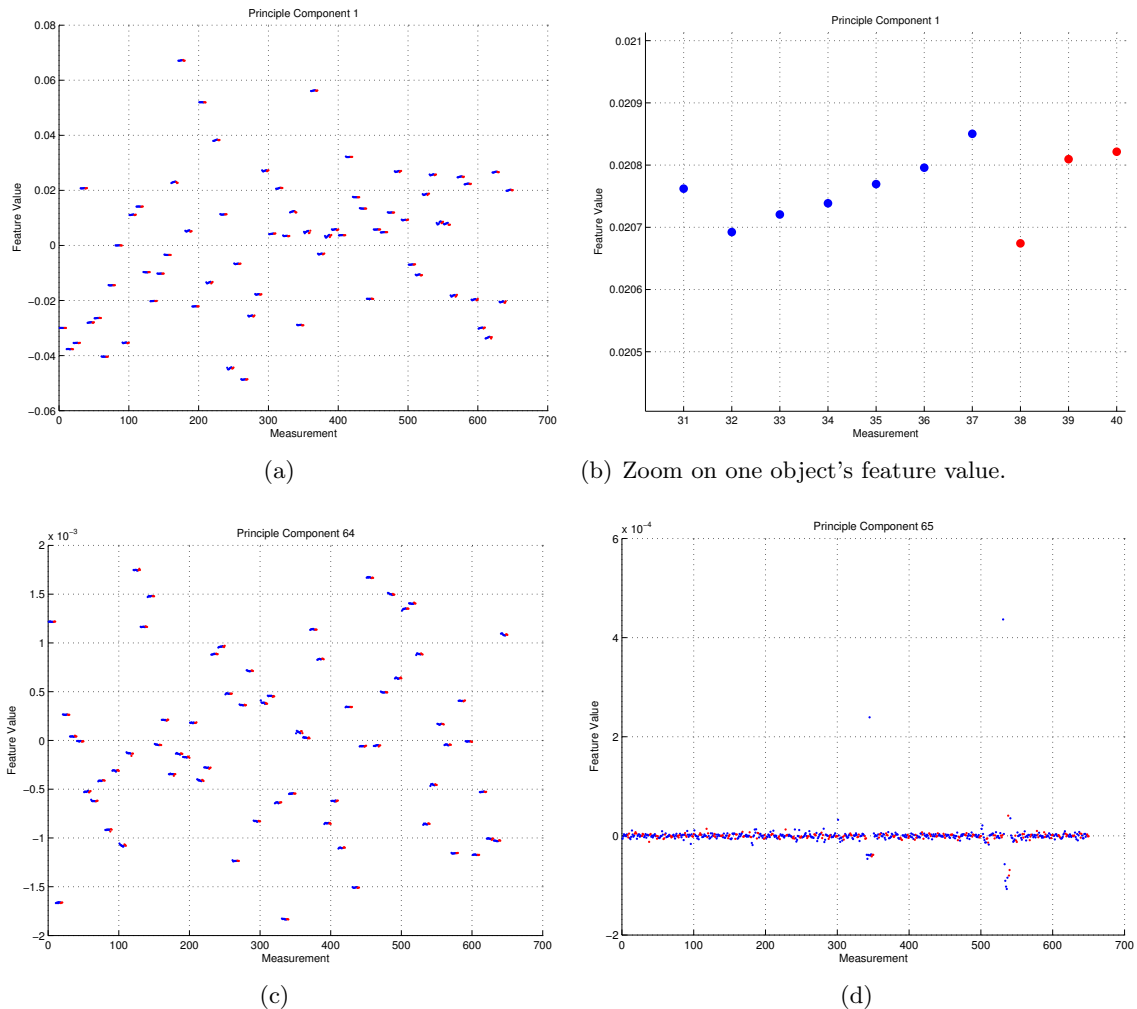


Figure 4.4.: Feature values of some principle components. Sweep range: 2.0–8.2MHz, number of points: 1601, objects: 65, measurements per object: 10 (7 train, 3 test).

the entropy (see section A.8.2 on page 72) of each principle component is calculated. The result is shown in figure 4.5. It can be concluded that the maximum number of feature values has not been reached yet and the figure predicts it is already possible to distinguish  $2^{428}$  delay

lines. In order to determine the maximum total entropy, more delay lines would be needed. But the limit could be quite high and more delay lines are not available.

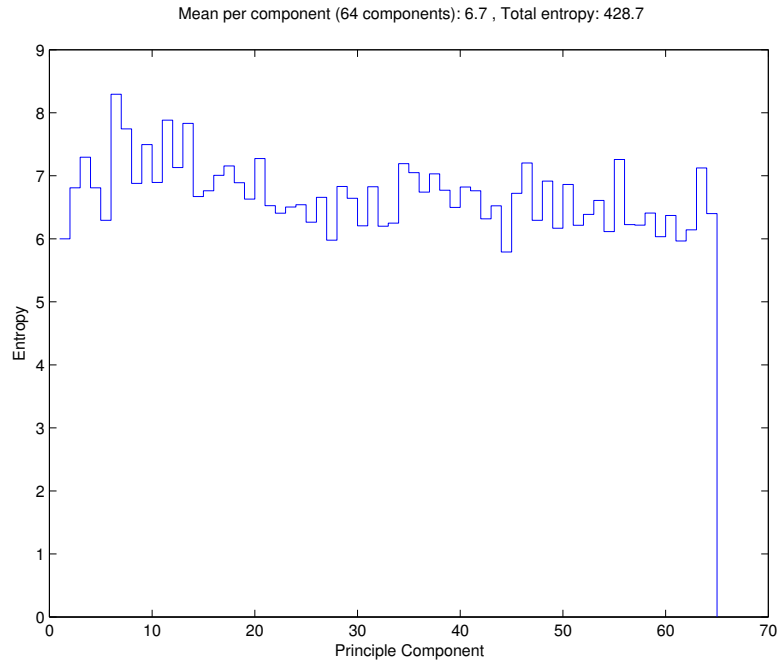


Figure 4.5.: Entropy of the principle components. Sweep range: 2.0 – 8.2MHz, number of points: 1601, objects: 65, measurements per object: 10 (7 train, 3 test).

### Reducing the Frequency Range

By reducing the frequency range and reducing the number of points, less entropy will be present in the principle components. Beyond that, it has the advantage that the time to make one measurement (and hence an identification) can be decreased. Figure 4.6 shows the entropy for measurements with the very small frequency range of about  $8 \cdot 8kHz$ . The entropy indeed decreases. By increasing/decreasing the number of points the entropy would be expected to increase/decrease. Table 4.6 shows some results. The table does not show what would be

Number of points	Principle components	Entropy	
		Total	Average
26	51	187.3	3.5
51	64	185.8	2.9
101	64	188.8	2.9
201	64	179.4	2.8
401	64	166.8	2.6

Table 4.6.: Entropy for different number of points.

expected. The trend is that with larger number of points, the entropy decreases. This can be explained by the fact that measurements with more points take longer to complete, which might introduce more noise in the results. Although the entropy of this small frequency range

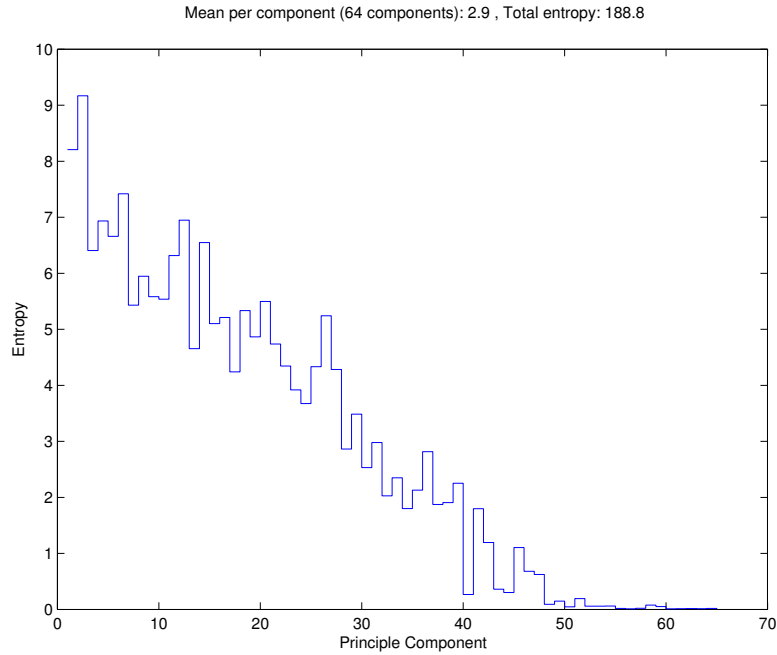


Figure 4.6.: Entropy of principle components. Sweep span:  $8 \cdot 8kHz$ , number of points: 101, objects: 65, measurements per object: 10 (7 train, 3 test).

might seem to be around 160 it is not possible to extrapolate these results to the full frequency range to obtain the total entropy in a DL701 since components in other frequency parts of the spectrum might be covariant with the frequency points in this part.

Figure 4.7 shows the logarithmic magnitude of the measurement for different settings for the number of points. At a certain number of points per frequency range, no information is added

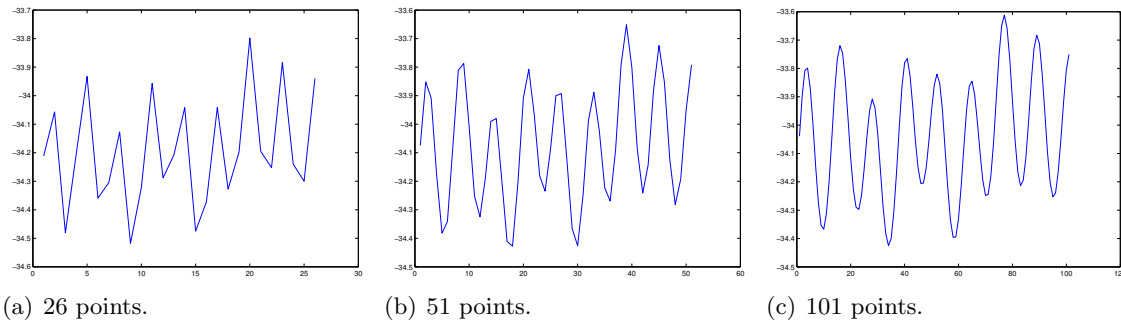


Figure 4.7.: LogMag for different number of points.

anymore by increasing the number of points. From figure 4.7 it can also be concluded that by measuring around 101 points in a frequency range of  $64kHz$  the transfer function on that frequency range can be fully recreated. This is important for exhaustive model building. Since the full frequency range that can be used for challenges is  $8.2 - 2 = 6.2MHz$ , this means that by measuring approximately  $(6.2 \cdot 10^6 / 64 \cdot 10^3) \cdot 101 = 9.8 \cdot 10^6$  frequency points, a DL701 can be fully modeled. Measuring all these points takes only  $(9.8 \cdot 10^6 / 1601) \cdot 0.8 \approx 4897$  seconds ( $\approx 82$  minutes). This means that if an original DL701 would be used as an APUF it is not protected



against an exhaustive modeling attack. If the transfer function was more unpredictable per small frequency step then the number of frequency points to measure and hence this time would increase. Another option would be to try to use slow materials (materials where the speed of sound is very low) as APUFs, but it is unlikely this will increase the time enough to make exhaustive modelling impossible. Although there could be solutions to the exhaustive modelling problem, it is strongly recommended to use control for a PUF anyway, even if the original PUF is strong. So the experiments are continued with the idea that control is added to the PUF.

#### 4.2.2. Demo System DL701 Identification

The AKI system incorporates a part to proof the concept of DL701 identification in real situations. It does not contain the code for identification enrollment and principle component extraction so this has to be done separately (see the AKI manual).

#### Results

This section shows some results obtained with the AKI system. Enrollment settings

Start frequency	2.0MHz
Stop frequency	8.2MHz
Nr. of points	51
Measurements per object	10 (7 train, 3 test)
Enrolled delay lines	DL00...DL27

Figure 4.8(a) shows an example of a distance graph that is shown on the screen during auto identification (at room temperature). From the graph it is clear that the distance (see equation 4.8) between one DL and the others is large enough to uniquely identify the DL701s. The horizontal line indicates the distance threshold AKI uses to identify DL701s, above the threshold, the inserted object is not identified. When all graphs are collected by inserting each object, figure 4.8(b) is obtained, which shows the distances of all the inserted objects.

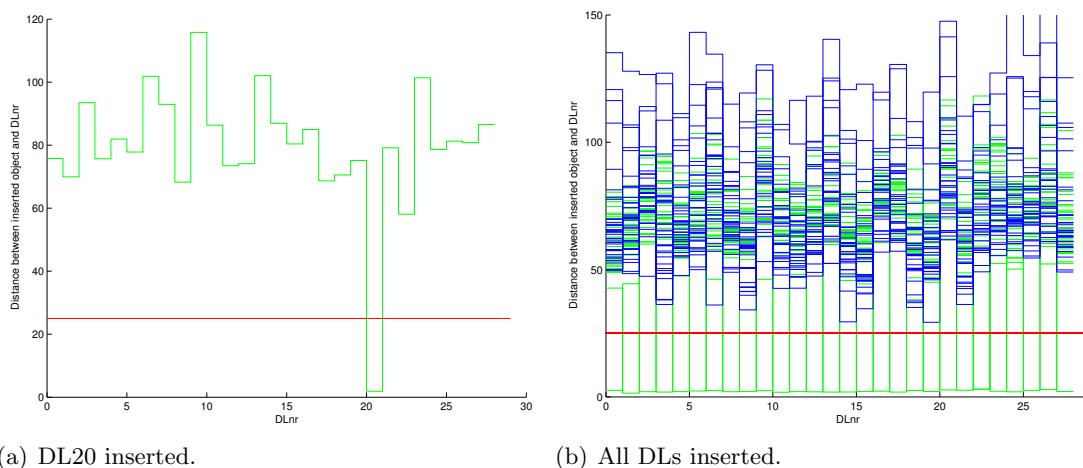


Figure 4.8.: Distance between inserted objects and stored data. Enrolled objects in green (light) unenrolled in blue (dark).

DLnr	Distance				DLnr	Distance	
	to itself		min. to $\neq$ itself			Day1	Day2
	Day1	Day2	Day1	Day2			
00	2.5741	1.3131	44.4176	44.4926	28	43.4199	43.5337
01	1.4629	1.9273	42.7267	42.2279	30	45.4680	45.0720
02	2.0289	1.6380	62.1344	62.1916	31	34.1801	34.0626
03	1.8877	2.0270	54.3735	54.7998	32	42.6266	43.1882
04	2.0681	2.4585	50.1089	50.4092	33	44.6083	45.0116
05	2.3374	2.2781	56.2276	56.7956	34	47.7220	46.6326
06	2.4608	2.2522	46.1818	46.1715	35	36.0951	35.5332
07	2.0258	1.6052	57.2867	57.1998	36	29.4698	29.4016
08	2.1408	2.2435	38.1092	38.3317	37	62.0707	62.3022
09	2.4267	1.9974	46.1934	46.2443	38	45.4171	44.6160
10	1.8373	1.7443	48.1156	47.7065	39	56.1793	55.8400
11	2.0785	1.9505	55.6053	55.4697	40	47.1726	46.4019
12	2.0194	1.7850	59.3003	59.8302	41	37.6588	38.1991
13	1.9051	2.0667	53.2844	53.5692	42	38.0638	37.0332
14	1.9483	1.8279	44.3764	45.0497	43	34.6462	34.7929
15	2.1484	2.0244	45.0342	45.4228	44	38.2991	38.0009
16	2.2589	2.1181	53.3652	53.5623	45	47.2764	47.9545
17	1.9146	1.9934	60.8770	61.0289	46	46.4390	46.2168
18	2.4885	2.5502	43.0610	43.1794	47	40.4419	40.1664
19	2.1256	2.2318	46.7574	46.9803	48	42.9670	43.0411
20	2.7512	2.0031	59.2448	58.6942	49	36.3065	35.8798
21	2.6244	2.1906	46.9419	46.6850	50	39.9252	40.3389
22	3.0124	2.4867	57.1993	57.1496	51	36.2962	36.2624
23	2.2272	2.2760	50.1786	50.3960	52	40.3574	40.1183
24	2.0218	1.6973	49.2927	48.8532	53	77.1662	76.5334
25	2.5792	2.3420	51.8319	51.5774	54	56.5335	50.0435
26	2.8773	2.8817	52.7138	52.2094	55	37.0267	56.5106
27	2.0967	1.9365	48.8165	48.9114			

Table 4.7.: Distances of inserted objects.

Table 4.7 shows some distances in a table. In the table the difference between one objects distance to its own data and the minimum distance that is found by comparing it to all other objects is shown. Note that delay lines 28 – 57 were not enrolled. If the values are assumed to be normally distributed, and two pdfs of the enrolled and unenrolled objects are made, figure 4.9 is obtained. It can be calculated that the two pdfs intersect at a distance of 3.82. Below some values for the *False Acceptance Rate* (FAR, distance of unenrolled objects  $\leq$  threshold) and *False Rejection Rate* (FRR, distance of enrolled objects  $>$  threshold) are shown for different thresholds.

Threshold	FRR	FAR
3.42	$1.10 \cdot 10^{-4}$	$1.68 \cdot 10^{-5}$
3.82 (intersection)	$6.06 \cdot 10^{-7}$	$2.00 \cdot 10^{-5}$
15	$0.00 * 10^0$ (matlab output)	$1.52 \cdot 10^{-3}$

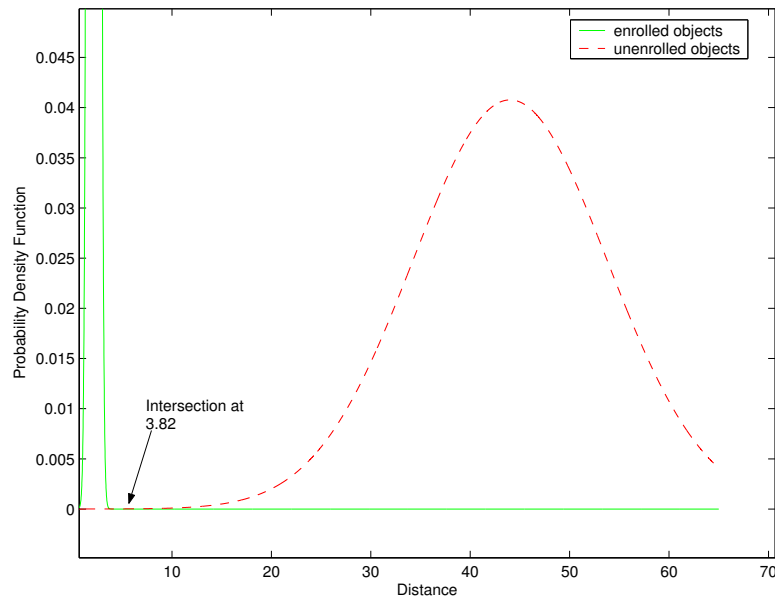


Figure 4.9.: Intersection of the pdfs.

The same can be done for other enrollment settings

Start frequency	4.4MHz
Stop frequency	4.432MHz
Nr. of points	51
Measurements per object	10 (7 train, 3 test)
Enrolled delay lines	DL00...DL27

Threshold	FRR	FAR
3.91	$1.01 \cdot 10^{-4}$	$2.26 \cdot 10^{-6}$
4.54 (intersection)	$5.45 \cdot 10^{-7}$	$4.58 \cdot 10^{-6}$
15	$0.00 * 10^0$ (matlab output)	$2.59 \cdot 10^{-2}$

Note that these values are valid if the measurements are done at room temperature (about  $20^{\circ}\text{C}$ ).

### Temperature Influences Identification

When DL701s are warmed up, the extracted feature values change, which also changes the distance measure. Table 4.8 shows an example of the change in the distance by warming a DL20 between two hands. From the table it is seen that warming increases the distance to a DL's own data. This also influences the FAR and FRR if the enrollment data is kept the same. On a cold day (unfortunately no temperature was recorded) the FRR can get as high as  $4.79 \cdot 10^{-3}$  and the FAR  $1.77 \cdot 10^{-2}$  if the threshold is set to the pdfs intersection ( $=12.14$ ). The influence of heat and other factors (moist etc.) should be further investigated. The change of feature values may be an obstacle for unique identification, but could also be a possibility to enlarge the challenge response space if the change in the feature values is unpredictable. It could be possible to fit a small heating/cooling element (e.g. peltier) inside the DL701 that

Measurement nr.	Distance	Distance to others
1	2.6978	59.3679
2	3.7287	56.0419
3	8.2475	49.9556
4	10.9121	52.5353
5	12.1144	51.0998

Table 4.8.: Warming DL20.

sets the temperature for a challenge [1, 8]. If the change in feature values *is* predictable, then a temperature sensor could be fitted to obtain the temperature to correct for the change in feature values.

### 4.2.3. Conclusion

DL701s can be uniquely identified quite easily when temperature is not varying too much and AKI is able to show this in practice. If a FRR is allowed of 1 out of 10,000 a FAR of 1 out of 100,000 should be easily possible, especially if temperature control will be used. The challenge space is not big enough to prevent exhaustive model building, but it is preferred to add control to a PUF anyway, which can prevent exhaustive model building.

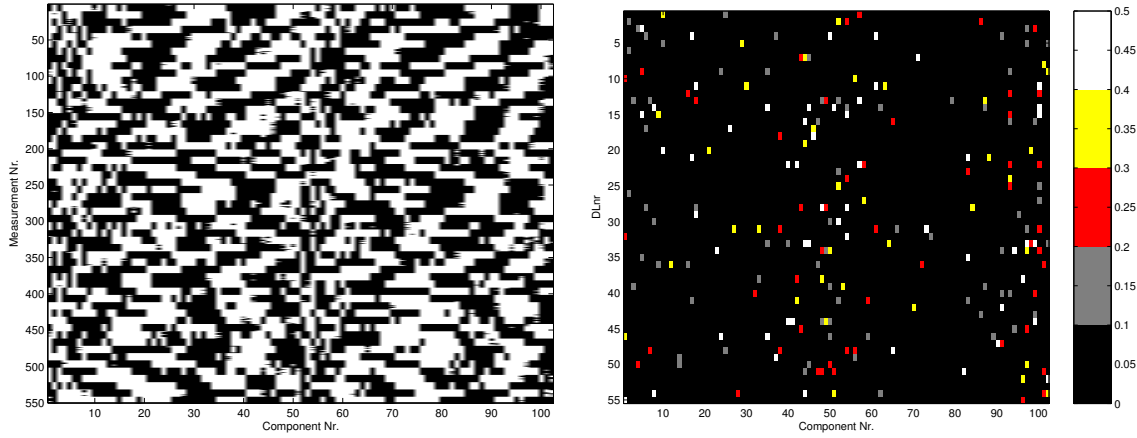
## 4.3. Threshold Key Generation

This section investigates whether key generation as described in section 1.6.1 on page 12 is possible for DL701s. Note that in order to truly compare DL701s no confusion or BCH error correcting code is added. Enrollment settings

Start frequency	2.0MHz
Stop frequency	8.2MHz
Nr. of points	51
Measurements per object	10 (7 train, 3 test)
Key generated for DL00...DL54	

Figure 4.10 shows the uncorrected keys and the flip rates that are generated by AKI. Flip rates indicate the number of times a bit changes (flips) during key extraction from enrollment measurements. Some of the components in some objects are not always converted to the same key and hence these components are tagged ‘unsteady’ with the help of the flip rates. The mean number of steady bits lies around 98, minimum 93, maximum 102 (of the 102 bits) for this example. The Fractional Hamming Distance (FHD) is calculated between all generated keys. The FHD is the number of components (in this case bits) that differ between two keys divided by the length of the key (number of components). This can be done on keys where the unsteady bits are removed and the keys are all made the same length or on keys where the unsteady bits are set to 1. For 55 objects  $\binom{55}{2} = \frac{55 \cdot 54}{2} = 1485$  FHDs are calculated.

Figure 4.11 shows a normalized histogram of the values obtained when unsteady bits are first removed. The mean of this distribution is 0.504240, the median 0.505376 and the standard deviation ( $\sigma$ ) 0.121759. If every bit in the key would be independent, it would be expected this histogram would have a binomial distribution with  $N = 93$  and  $p = 0.5$ . Where the



(a) Generated keys without unsteady bit correction. (b) Derived fliprate.  
First ten measurements of DL00, last ten of DL55.

Figure 4.10.: Uncorrected keys and flip rate.

number of 1's in each key is taken as the number of successes in 93 draws. If this distribution is drawn in the same figure, it does not match the experimental data. This could be because the bits in the key are not all independent. The *effective* number of independent bits can be determined by the looking at the experimental data according to [22]. For a binomial distribution  $f$ ,  $var(f) = \sigma^2 = N \cdot p(1 - p)$ . Since the FHD is calculated (and not the HD), the variance of the FHD distribution is  $\sigma^2 = N \cdot p(1 - p)/N^2$ , which means  $N = p(1 - p)/\sigma^2$ . For  $p = 0.504240$  and  $\sigma^2 = 0.121759^2$ ,  $N_{effective} \approx 17bits$ . This observed distribution is also plotted in the same figure and it can be seen that it matches the experimental data. Some results obtained

Frequency [MHz]		Nr. of Points	Unsteady Removed?	Bits in Key	Mean FHD	$\sigma$	Effective Bits
Start	Stop						
2	8.2	51	No	102	0.503387	0.135958	14
2	8.2	51	Yes	93	0.504240	0.121759	17
2	8.2	1601	No	3202	0.504019	0.137818	13
2	8.2	1601	Yes	3141	0.500517	0.052861	89

It seems strange that if unsteady bits are removed the number of independent bits increases. This can be explained by the fact that the other bits in keys are shift when bits are removed, this could introduce more randomness as originally present. Since the information which bits to remove is public, it does not add more randomness.

To guarantee the keys are distinct AKI adds confusion (as described in 1.6 on page 11). Adding random confusion makes the keys more different in case they are related, but does not reduce the FAR when processing takes place outside the PUF.

To correct for small errors in the derived key and to add confusion, AKI uses a small BCH error correcting code. For keys of 102 bits, the code uses codewords of 127 bits (N) and can correct 4 bits (T) to extract a message of 99 bits (K). When more errors are present in the

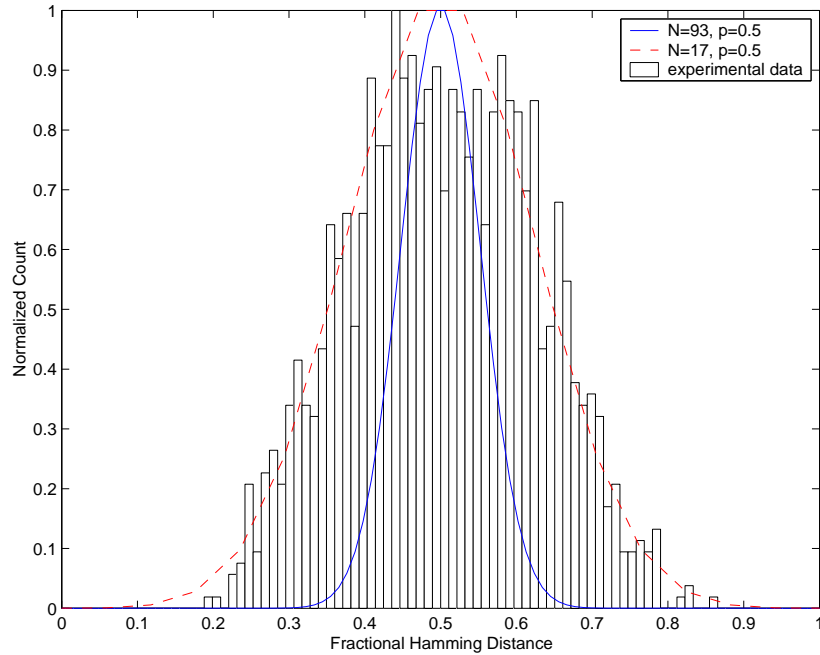


Figure 4.11.: Experimental and theoretical binomial distribution.

derived key, a code that can correct for more bits has to be used e.g.  $N = 255, K = 99, T = 23$ . This correction however, also increases the FAR when processing takes place outside the PUF, since keys which differ in 23 bits can now be error corrected to the same keys. Further, the derived key is 102 bits long and must be XOR-ed with the codeword of 255 bits (by adding 0's). This leads to more information leakage about the secret contained in the codeword.

#### 4.3.1. Conclusion

The detection of unsteady bits can probably be made much better by defining a small region around the threshold, based on the standard deviation in each component. Measurement values within this region are then tagged as unsteady. The effective number of bits with this method is too small to generate useful keys from DLs. When processing takes place outside the PUF, adding an error correcting code increases the FAR, because different keys have a higher chance of resulting in the same secret. Another reason to aim for controlled PUFs where processing can take place inside the PUF. Nevertheless, adding error correction inside the PUF, makes opening the PUF and generating a fake measurement signal from the PUF easier, since more (fake) signals will result in the same secret.

## 4.4. Quantized Key Generation

This section investigates whether key generation as described in section 1.6.2 on page 14 is possible for DL701s. In order to aid in choosing the quantization, AKI presents some figures during quantized key extraction (see the manual in section C). First, the quantization is done to only two values ( $v = 2 \equiv \{0, 1\}$ ). With the following enrollment settings

Start frequency 2.0MHz  
 Stop frequency 8.2MHz  
 Nr. of points 30, randomly chosen from the range  
 Measurements per object 10 (7 train, 3 test)  
 Quantization chosen  $q = \text{factor} * \sigma_n$ , where n is the noise in each component  
 factor = 8  
 Key generated for DL00...DL10

The resulting FHDs are shown in figure 4.12 Compared to the threshold key extraction

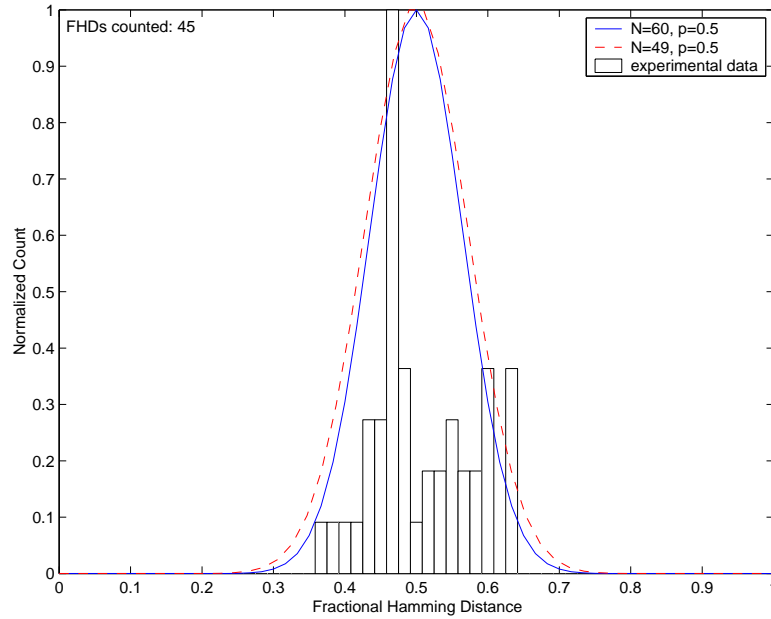


Figure 4.12.: Experimental and theoretical binomial distribution for QKE.

method, the effective number of bits is much higher. Ten keys were extracted, so  $N_{effective}$  is based on 45 ( $= 10 * 9/2$ ) comparisons. Note that the histogram should match the calculated binomial distribution when more keys are derived, but in order to calculate the effective number of bits 45 samples is enough to roughly estimate  $\sigma^2$  and hence the effective number of bits. A different data set, with the same size gave  $N_{effective} = 45$ . From figure 4.13 one can see that the extracted keys look more random than with threshold key extraction. Figure

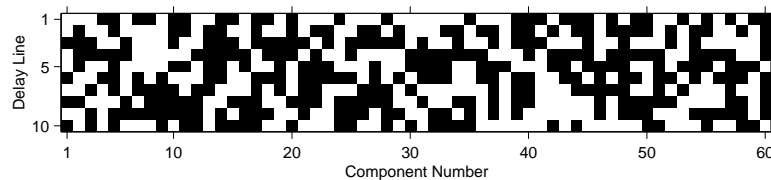


Figure 4.13.: Keys generated with quantized key extraction.

4.14 shows the components contain much more entropy than 1 bit and the number of bits can be enlarged by quantizing to multiple values. AKI quantizes standard to  $v = 2^{\text{floor}(\text{entropy})}$  values for each component. More values is useless since the components do not contain more

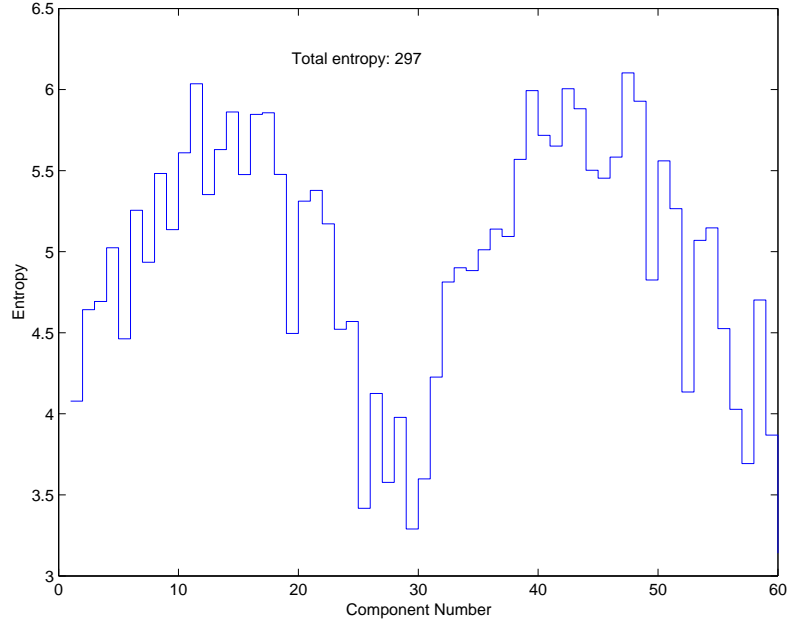


Figure 4.14.: Entropy of components.

entropy. An example for the keys generated like that is shown in figure 4.15. From the figure it is clear that the component values are spread over the range of available values, so the keys are all very different. When quantizing to multiple values it is not possible to calculate the effective bits immediately. The FHDs multi valued keys can be determined and are shown in figure 4.16. The figure also shows the *normalized distances*. Which are defined as follows

$$D_{component\ ij\ k} = MIN(v - |key_{i\ k} - key_{j\ k}|, |key_{i\ k} - key_{j\ k}|) \quad (4.19)$$

$$D_{normalized\ ij} = \frac{1}{c} \sum_{k=1}^c \left( \frac{D_{component\ ij\ k}}{v/2} \right) \quad (4.20)$$

with:

- $D_{component\ ij\ k}$  : Distance between component k of keys i and j
- $key_{i\ k}$  : Value of component k of key i
- $v$  : Number of key values that is quantized to
- $c$  : Number of components in a key
- $D_{normalized\ ij}$  : Distance measure between key i and j

This means if keys are exactly  $v/2$  apart on each component, the normalized distance would be the maximum of 1. From the figure 4.16 one can conclude that the keys are indeed clearly distinguishable when looking at the FHDs.

The number of effective bits can be calculated by first converting the decimal valued components to binary values. And use these (longer) bit keys to calculate the number of effective bits as was done in the previous sections. Figure 4.17 shows the results. This figure shows that



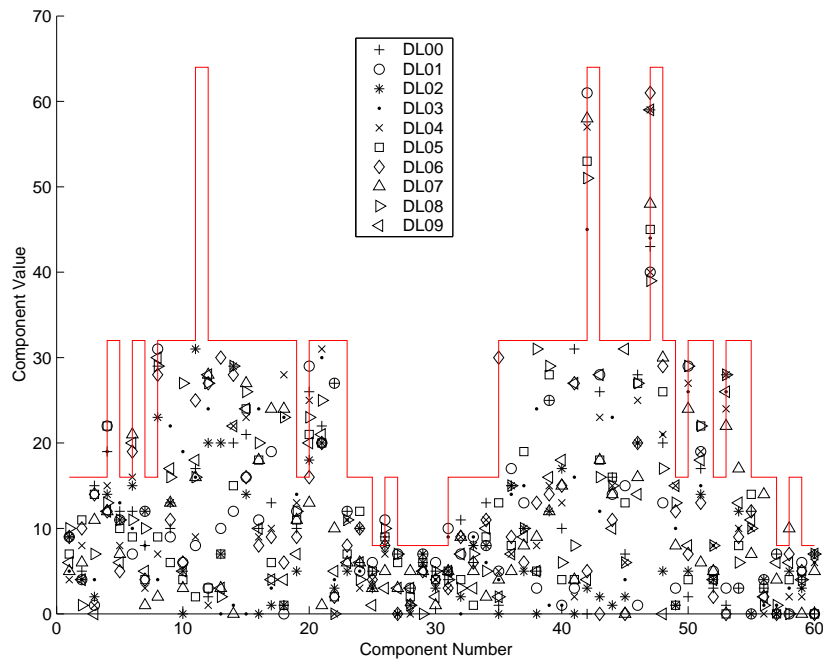
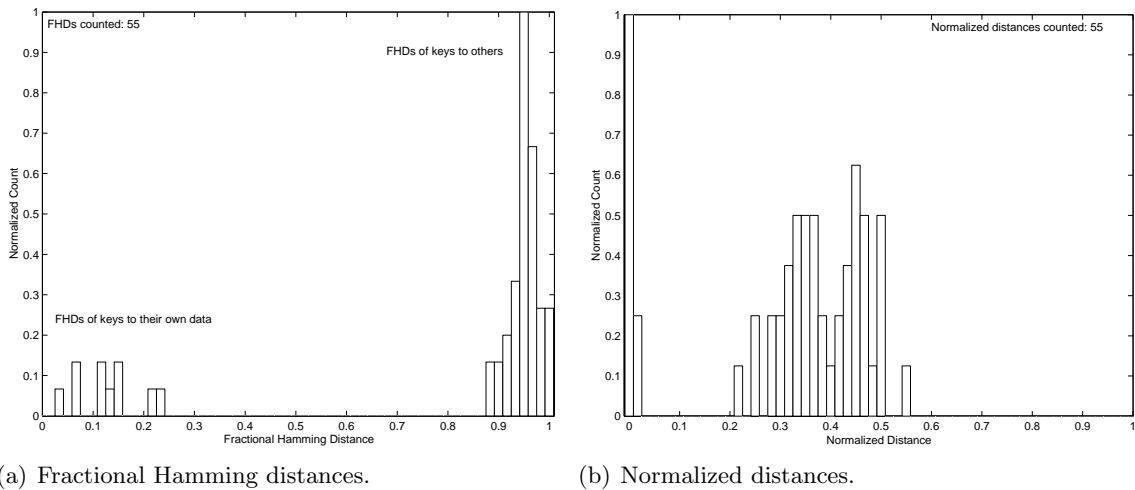


Figure 4.15.: Keys generated by quantized key extraction to multiple values. The red line indicates the  $v$  chosen for that component. Each marker belongs to the key values of one delay line as shown in the legend.



(a) Fractional Hamming distances.

(b) Normalized distances.

Figure 4.16.: Distances for QKE between keys.

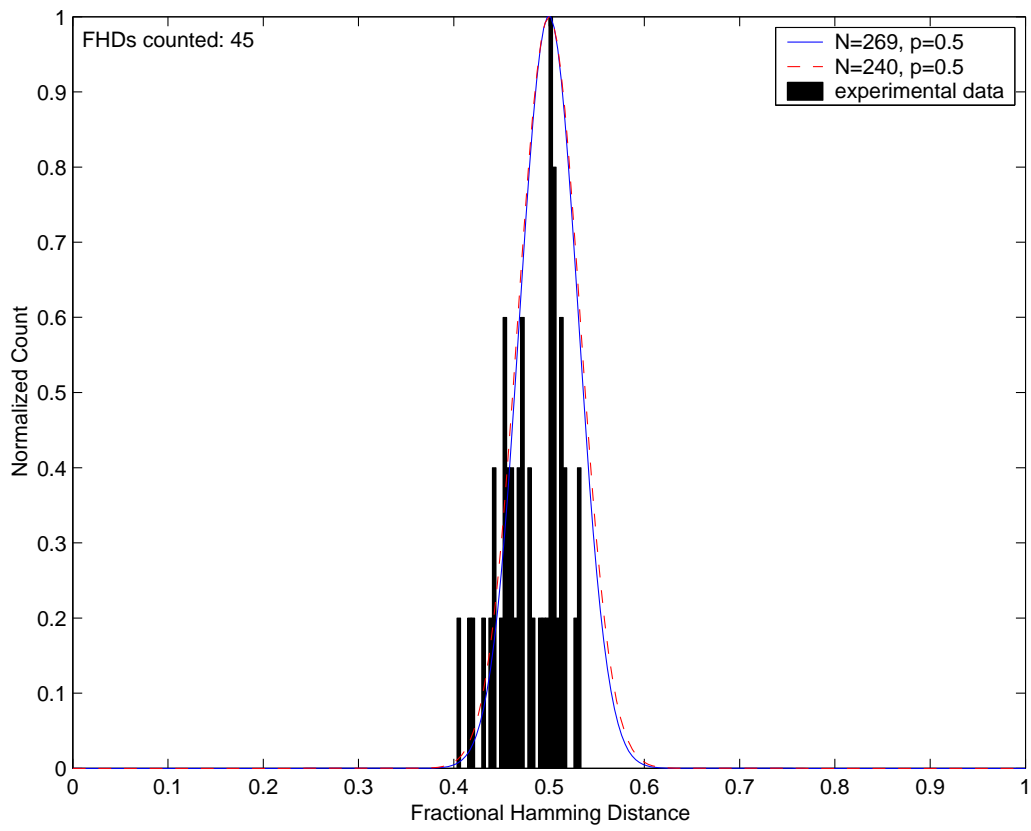


Figure 4.17.: Experimental and theoretical binomial distribution when decimal key values are converted to binary values.

the effective number of bits is 240 (FHD mean: 0.479802, FHD std: 0.032237). This is clearly a big improvement! Note however that one should take into consideration that by choosing  $v$  higher, more information is leaked from the helper data when helper data is used (see section 1.6.2). Always valid is  $\sigma_x/\sigma_n = \sqrt{2^{2H} - 1}$ , which means  $\sigma_x = \sqrt{2^{2H} - 1} \cdot \sigma_n$  (equation A.24). The probability of error in one bit is dependent on  $\frac{q}{\sigma_n} = \frac{\text{factor} \cdot \sigma_n}{\sigma_n} = \text{factor}$  (see figure 1.8). Because the error probability should not be too high ( $\leq 10^{-4}$ ), ‘factor’ is chosen 8. Information leakage is dependent on  $\frac{\sigma_x}{q} = \frac{\sqrt{2^{2H} - 1} \cdot \sigma_n}{\text{factor} \cdot \sigma_n} = 1/8 \cdot \sqrt{2^{2H} - 1}$  (see figure 1.11).  $\sigma_x/q$  is calculated for several values of  $H$

$H$	$\sigma_x/q$	$H$	$\sigma_x/q$
1	0.22	4	2.00
2	0.48	5	4.00
3	0.99	6	8.00

Most of the components have an entropy of  $\geq 4$ , so  $\sigma_x/q \geq 2$ . In the ‘information leakage figure’,  $v$  can be chosen, depending on the amount of leakage allowed. For  $v \leq 4$  information leakage stays below  $10^{-4}$  per component. If the components are converted to binary values and the effective number of bits is calculated for this setting,  $N_{effective} = 132$  bits is obtained. This cannot be true, since the bit keys are only 120 bits long after conversion (2 bits needed per component). Probably too little keys were extracted to make a very accurate estimation of variance of the FHD distribution (which is used to calculate  $N_{effective}$ ).

#### 4.4.1. Conclusion

Quantized key extraction seems the most effective and reliable method to extract keys from DL701s. There still needs to be implemented a efficient way to correct for errors in the keys. This could for example be done by converting the decimal key values to bits and using an error correction code on them, but there is probably better error correction method for multiple valued data (e.g. Reed-Solomon codes processes symbols in  $2^m$  instead of 2). Further, more keys should be extracted to verify the results in the last section and make a more accurate estimation of  $N_{effective}$ .

## 4.5. PUF on chip Experiment

As described in section 1.4, controlled PUFs are very interesting to build. In order to investigate whether it is possible to make an APUF on chip, measurements on an DL701 were done before and after it was covered with epoxy material. Chips normally are poured in epoxy material in the final creation process. If an APUF would be made on chip, it could be possible that the epoxy cover could damp out all the measurement energy, so the output signal (response) could become unmeasurable. The experiment indeed confirmed this: no signal could be measured anymore with the epoxy material poured in. Other methods and other materials as APUF could make a CPUF possible, but this has to be investigated.



## 5. Conclusion

DL701s can be uniquely identified quite easily when temperature is not varying too much and AKI is able to show this in practice. If a FRR is allowed of 1 in 10,000 a FAR of 1 in 100,000 should be easily possible, especially if temperature control will be used. The challenge space is not big enough to prevent exhaustive model building, but it is preferred to add control to a PUF anyway, which can prevent exhaustive model building.

The effective number of bits generated by using threshold key extraction to extract keys from delay lines, is too small to generate useful keys from DLs.

However, quantized key extraction used to extract keys from DL701s clearly shows Acoustical PUFs are feasible. From 30 measurement points (60 components) it is possible to generate a key of about 120 bits (Note: these experiments should be repeated for more extracted keys, to make a more accurate estimation). When the key that is directly generated by the PUF is immediately used as a key (so no confusion is added) then this number of bits is probably even higher (in the order of 240 bits).

Since exhaustive modeling is very likely, best thing is to go for controlled PUFs. Controlled PUFs have many advantages over normal PUFs.

So, APUFs look promising, but there is still a lot of matters to investigate. These matters are discussed below.

### 5.1. Future work

- As stated in section 1.2 on page 5 more research to modelling of piezoelectric transducers and waves in solids should be done in order to investigate how hard adaptive model building is.
- Can they be simulated easily? What is the physics behind APUFs and is it possible to create a model that can predict their responses?
- How uncloneable are APUFs specifically? Can APUFs be created which are even more difficult to clone?
- Save the difference between succeeding mini-responses as response to a challenge. In AKI3 the actual values are saved and a reference challenge is always used to account for shifts of the values during time. Since only the mean of the real and the mean of the imaginary part is used for shifting, probably the differences don't change. Saving the differences could have the following advantage:
  - reference challenge may not be necessary any more. Note that one should distinguish between the real part of the response and the imaginary part: don't save the

difference between the two neighboring points that each lie in the different parts (in the middle of the response).

- one could use the reference challenge for auto identification: the PUF is inserted and the system recognizes which PUF it is. The reference response is used to search the database for the response
- Investigate change of feature values (for identification) in relation to temperature. Is the change predictable?

Some future work that could be done on AKI (APUF Key extraction & Identification):

- Expand AKI (APUF Key extraction & Identification) to:
  - show encryption/decryption of files
  - support multiple challenge/response pairs per APUF
- Rewrite AKI to clean up the code.
- Find more information about the work of Otto Muskens. During the finishing of this Report the following information was found in an old Cursor (see [21], note this is in Dutch)



These very short pulses could perhaps be used as a new type of challenge to create PUFs. More on his research can be found on <http://www.phys.uu.nl/~muskens/SolitonProject.php>.

## Bibliography

- [1] An Introduction to Thermoelectric Coolers — Electronic Cooling. Avail: [http://www.electronics-cooling.com/Resources/EC\\_Articles/SEP96/sep96\\_04.htm](http://www.electronics-cooling.com/Resources/EC_Articles/SEP96/sep96_04.htm).
- [2] Birthday Attack from Mathworld — Wolfram Research. Avail: <http://mathworld.wolfram.com/BirthdayProblem.html>.
- [3] Channel Coding. Avail: [http://cwww.ee.nctu.edu.tw/course/channel\\_coding/chap5.pdf](http://cwww.ee.nctu.edu.tw/course/channel_coding/chap5.pdf).
- [4] Connexions — Sharing Knowledge and Building Communities. Avail: <http://cnx.rice.edu/content/m10180/latest/>.
- [5] GSM World from the GSM Association — GSM Algorithms Subject to Application. Avail: <http://www.gsmworld.com/using/algorithms/index.shtml>.
- [6] Introduction to Error Correcting Codes. Avail: <http://web.media.mit.edu/~kung/publication/ecc.pdf>.
- [7] NDT Education Center. Avail: <http://www.ndt-ed.org/EducationResources/CommunityCollege/Ultrasonics/Physics/>.
- [8] Peltier Elements — Eureka Messtechnik. Avail: <http://www.eureca.de/english/peltierelement/science.html>.
- [9] Principle Components Analysis (PCA) — Tutorial compiled by a Biochemistry Research Team from the University College London. Avail: [http://www.ucl.ac.uk/oncology/MicroCore/HTML\\_resource/tut\\_frameset.htm](http://www.ucl.ac.uk/oncology/MicroCore/HTML_resource/tut_frameset.htm).
- [10] Passive Components Produkt Programme — Philips Components. Philips Nederland BV, Marktgroep Philips Components, 1993.
- [11] Elad Barkan, Eli Biham, and Nathan Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. Technical report, Technion — Israel Institute of Technology, Haifa 32000, Israel, May 2003. Avail: <http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2003/CS/CS-2003-05.pdf>.
- [12] Marc Briceno, Ian Goldberg, and David Wagner. GSM Cloning, April 1998. Avail: <http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html>.
- [13] C.F. Brockelsby and J.S. Palfreeman. Ultrasonie Verdragingslijnen en hun Toepassing in de Televisie. In *Philips Technisch Tijdschrift*, number 10 in Jaargang 25, pages 326–347. Philips, 1963.

- 
- [14] Donald. E., S.D. Crocker, and J.I. Schiller. RFC 1750 — Randomness Recommendations for Security, December 1994. Avail: <http://www.faqs.org/rfcs/rfc1750.html>.
- [15] Blaise L. P. Gassend. Physical random functions. Master's thesis, Massachusetts Institute of Technology, Februari 2003.
- [16] Blaise L. P. Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Controlled Physical Random Functions. Technical report, Massachusetts Institute of Technology, Cambridge, USA, 2002.
- [17] Blaise L. P. Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicone Physical Random Functions. Technical report, Massachusetts Institute of Technology, Cambridge, USA, 2002.
- [18] Hideki Imai et al. CRYPTREC Report 2002. Technical report, Information-technology Promotion Agency, Japan. Telecommunications Advancement Organization of Japan, March 2003.
- [19] Jean-Paul Linnartz and Pim Tuyls. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. Technical report, Philips Research Laboratories, 2003.
- [20] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press Inc., electronic edition, 1997.
- [21] Otto Muskens. Geluidspuls in kristal. Avail: [http://www.tue.nl/cursor/bastiaan/jaargang46/cursor26/achtergrond/k\\_o.html](http://www.tue.nl/cursor/bastiaan/jaargang46/cursor26/achtergrond/k_o.html).
- [22] Pappu Srinivasa Ravikanth. *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology, March 2001.
- [23] Pappu Srinivasa Ravikanth, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One-Way Functions. *Science*, 297:2026–2030, September 2002.
- [24] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., second edition, 1996.
- [25] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948. Avail: <http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf>.
- [26] B. Škorić, Pim Tuyls, Sjoerd Stallinga, Ton Akkermans, and Wil Oprey. An Information-Theoretic Model for Physical Unccloneable Functions. Technical report, Philips Research Laboratories, 2003.
- [27] B. Škorić, Pim Tuyls, Sjoerd Stallinga, Ton Akkermans, and Wil Oprey. Optical PUFs. Technical report, Philips Research Laboratories, October 2003. Unpublished.
- [28] A. L. Zijlstra and C. M. van der Burgt. Isopaustic Glasses for Ultrasonic Delay Lines in Colour Television Receivers and in Digital Applications. *Ultrasonics*, January 1967.



## A. Cryptography

In order to compare the characteristics of *Physical Unclonable Functions* (PUFs) with classical cryptographic systems some general cryptographic terms are explained in this chapter. The last section will discuss the entropy a measured value contains. This is related to cryptography when for example measurement values are converted to an encryption key.

### A.1. Symmetric Encryption Algorithms

#### A.1.1. One-time pad

A one-time pad (Vernam cipher) uses a string of bits that is completely random (generated from a true random source) to encrypt plaintext. Encryption is done by combining the one-time pad with the plaintext (usually done with the bitwise XOR). Since the entire keystream is random, an attacker with infinite computational resources can never know the plaintext if she only sees the ciphertext because all plaintexts will have equal probability.

#### A.1.2. Stream/Block Ciphers

Since one-time pads are often not really practical, stream ciphers were developed. The difference with the one time pad is that their output is deterministic and depends on an internal state. While stream ciphers are unable to provide the same theoretical security of the one-time pad, they are at least practical. Stream ciphers generate keystreams and encryption is done by combining the keystream with the plaintext (bitwise XOR). There are two types of stream ciphers:

- Synchronous - the generation of the keystream is independent of the plaintext and ciphertext.
- Self-synchronizing - the keystream is dependent on a fixed number of bits from the ciphertext and/or plaintext.

Stream ciphers work on the plaintext one single bit (or byte) at a time. Ciphers that work on the plaintext in groups of bits of a fixed length (block), are called block ciphers. Block ciphers used in a certain mode can be transformed into a stream cipher and hence any block cipher can be used as a stream cipher, but dedicated designs are usually faster (and more secure). By construction, the sequence generated by stream- and block ciphers will eventually repeat itself. This is another important difference with the one time pad, which does not have a period of repetition.

### A.2. Perfect Secrecy

A cipher is *perfect* if for every cryptogram  $y^N$  and plaintext  $x^N$  (where  $N$  is the length of the plaintext) the a posteriori probability (the probability of occurrence, for the given information

and observations) of both is equal. Then a cryptanalyst does not receive any information about the plaintext. This condition implies that the plaintext and the cryptogram have to be *statistically independent*. To determine the ‘uncertainty’ over the set of plaintexts  $X^N$ , the entropy is used. Entropy is the measure of the disorder or randomness of a system. Figure A.1 summarizes some entropy relations. Statistical independence (perfect secrecy) can be written as

$$H(X^N|Y^N) = H(X^N)$$

with:

$H(A|B)$  : Entropy of  $A$  given  $B$   
 $H(A, B|C)$  : Entropy of  $A$  and  $B$ , given  $C$   
 $N$  : Length of the plaintext  
 $X$  : Set of plaintexts  
 $Y$  : Set of ciphertexts

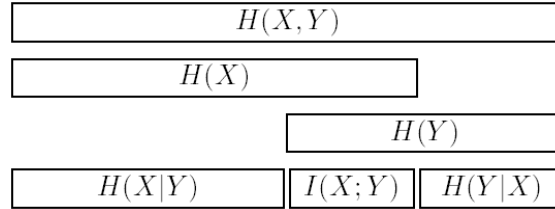


Figure A.1.: Some entropy relations,  $I(X; Y)$  : Mutual information between  $X$  and  $Y$ .

Since  $Y^N$  can be calculated from  $Z$  (the set of keys) and  $X^N$  and likewise  $X^N$  can be calculated from  $Z$  and  $Y^N$  it is known that

$$H(Y^N|X^N, Z) = 0 \quad (\text{A.1})$$

$$H(X^N|Y^N, Z) = 0 \quad (\text{A.2})$$

Given the entropies above the following can be derived

$$\begin{aligned} H(X^N, Z|Y^N) &= H(X^N|Y^N) + H(Z|X^N, Y^N) \\ &\geq H(X^N|Y^N) \end{aligned} \quad (\text{A.3})$$

$$\begin{aligned} H(X^N, Z|Y^N) &= H(Z|Y^N) + H(X^N|Y^N, Z) \\ &= \{\text{use eq. A.2}\} \\ &\quad H(Z|Y^N) \\ &\leq H(Z) \end{aligned} \quad (\text{A.4})$$

$$\text{from eq. A.3 and A.4} \Rightarrow H(X^N|Y^N) \leq H(Z)$$

Thus for a system with perfect secrecy we conclude

$$\begin{aligned} H(X^N) &= H(X^N|Y^N) \\ &\leq H(Z) \end{aligned} \quad (\text{A.5})$$

So for perfect secrecy the key uncertainty  $H(Z)$  must be at least as large as the message uncertainty  $H(X^N)$ . For the entropy of the key and the plaintext the following applies (see also equation A.12 on page 72 with  $p_i$  substituted)

$$H(Z) \leq K \log_2 |\mathcal{Z}| \quad (\text{A.6})$$

with equality only if the key selection is completely random.

$$(p_i = \frac{1}{|\mathcal{Z}|^K})$$

$$H(X^N) \leq N \log_2 |\mathcal{X}| \quad (\text{A.7})$$

$$(p_i = \frac{1}{|\mathcal{X}|^N})$$

(A.8)

with:

- $K$  : Length of the key
- $N$  : Length of the plaintext
- $|\mathcal{Z}|$  : Key alphabet size
- $|\mathcal{X}|$  : Plaintext alphabet size

Now if  $|\mathcal{X}| = |\mathcal{Z}|$ , as in the one-time pad and the plaintext is completely random then  $K \geq N$ . So the key has to be at least as large as the plaintext, cf. the one-time pad (see section A.1.1 on page 63).

### A.3. Hash and one-way functions

In this section hash functions and one-way functions are described. They are fundamental primitives in cryptography [24, 20]. One primitive is the **one-way function**

**Definition 5** *A one way function  $h$  is a computationally efficient function with the following property:*

- *Given  $x$  it is easy to compute  $h(x)$ , but given  $h(x)$  it is infeasible to compute  $x$ .*

There is no proof that real one-way functions exist, but many functions look one-way: there is no known way to reverse them. One-way functions are no good for encrypting messages: it would be impossible to decrypt them again.

**Trapdoor one-way functions** (or keyed hash functions) are special one-way functions. They work the same except if one knows some secret information  $y$ , then with  $h(x)$  and  $y$  it is easy to compute  $x$ .

A **one-way hash function** is also called: compression function, contracting function, message digest (MD), fingerprint, cryptographic checksum, message integrity check (MIC) or manipulation detection code (MDC). One way hash functions are often used in cryptography.

**Definition 6** *A one-way hash function is a computationally efficient function, mapping binary strings of arbitrary length (pre-images) to binary strings of some fixed length (hash-values). With the following property:*

- Given  $x$  it is easy to compute  $h(x)$ , but given  $h(x)$  it is unfeasible to compute  $x$ .

A good one-way hash function is *collision free* and shows the *avalanche effect*: it is hard to generate pre-images with the same hash value and if a single bit changes in the pre-image, on average half of the bits in the hash-value change. The point is to produce a value that can be used to verify whether another message is likely to be the same as the message that produced the value. Note that hash functions can never realize 100% collision resistance because it permits a larger input bit length than the output bit length. For this reason, the hash function is considered to have collision resistance if no collision is detected within a realistic computational complexity. Table A.3 shows characteristics of some hash functions [18].

Algorithm	RIPEMD-160	SHA-1	SHA-256	SHA-384	SHA-512
Permitted length of input message [bits]	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Output hash length [bits]	160	160	256	384	512
Block length for each basic process unit [bits]	512	512	512	1,024	1,024
Word length for each basic operation processing [bits]	32	32	32	64	64
The number of processing steps	160	80	64	80	80

Table A.3.: Characteristics of some hash functions.

‘Birthday attacks’ are often used to find collisions of hash functions. For a function with a random input that returns one of  $k$  equally-likely values, one can repeatedly evaluate the function for different inputs. The same output is expected after about  $1.2 \cdot \sqrt{k}$  evaluations [2]. It gets its name from the surprising result that the probability that two or more people in a group of 23 ( $k = 365$ ) share the same birthday, is greater than 1/2.

**Message authentication codes** (MACs) or data authentication codes (DACs), are one-way hash functions that also use a secret key as input for the function. Only someone that knows the key can verify the hash-value.

## A.4. Asymmetric Encryption

While one-way functions are not useful for encryption, trapdoor one-way functions are. Asymmetric encryption is based on trapdoor one-way functions (see section A.3 on the preceding page) and is often called public-key cryptography. Encryption is the easy direction and done by using a public key. Decryption is the reverse and should not be possible without the secret (or often called private) key.

Public-key crypto systems are vulnerable to chosen plaintext attacks. An attacker can en-

crypt all  $n$  possible plaintexts (if this set is small) and compare it with the ciphertext. The attacker is not able to get the secret key this way, but is able to determine the plaintext. Symmetric crypto systems are not vulnerable to this attack because an attacker cannot perform trial encryptions with an unknown key.

## A.5. Digital Signatures

Digital signatures are used to bind an identity to a piece of (digital) information. Signing is the transformation of a message and some secret only known to the identity, into a tag: the *signature*. Signing can be done in several ways:

- By use of symmetric crypto systems and an arbitrator. The arbitrator has to be trusted by the communicating parties.
- Using digital signature trees, which contain an infinite number of one-time signatures.
- Using public-key cryptography to sign the document.
- Using Public-key cryptography and one-way hash functions to sign the one-way hash of the document. In practice public-key algorithms are too expensive (time, memory) to sign long documents. Hence signature protocols are often based on signing the one-way hash of a document. The use of one-way hash functions also makes it more easy to create multiple signatures on one document.

Digital signatures often include time stamps to prevent reuse of the signed document (like digital checks). However, the owner of the signature can always sign a document and afterwards deny the signing by claiming that her key was compromised. This is called repudiation and also the reason why there is much discussion about private keys in tamper-resistant modules. These should make sure the owner cannot get to the key and abuse it, for example by ‘loosing’ it accidentally.

## A.6. Identification of Users

Identification of a user is usually based on a shared secret or a biometric of the user.

- Based on user’s memory.
  - Password
  - One-time password
  - Challenge response method based on user’s memory
- Based on user’s belongings.
  - Magnetic cards
  - Smart Cards
  - Documents
  - Mobile phones
  - PDAs (Personal Digital Assistants)

To prevent forgery, tamper-resistant modules should be used that are clone-resistant.

- Based on user's biometrics
  - Fingerprints
  - Retinal prints
  - Iris prints
  - Voice prints
  - Facial profiles
  - Handwriting

The biometrics do not impose a severe burden on the users' memory or belongings, but usually contains private information.

## A.7. (Pseudo) Random Number Generators

Random number generators can generally be classified into two categories

**Random number generators** RNGs: non-deterministic random number generators. RNGs generate random numbers from a certain physical quantity (noise from an electrical circuit, Geiger counter clicks or quantum effect of a semiconductor). The RNG output can be used as a random number or as an input to a PRNG.

**Pseudo random number generators** PRNGs: deterministic random number generators. PRNGs are a mechanism for generating random numbers on a computer. A PRNG generates multiple 'pseudo-random numbers' for one or more inputs usually using a RNG output as a seed. It is called pseudo random, because truly random numbers cannot be generated from a completely non-random thing like a computer. Typically the same sequence of pseudo random numbers is generated when starting from the same seed. A PRNG has a secret state  $S$  and upon request it generates outputs that are indistinguishable from random numbers to someone that does not know  $S$ . Hence it looks very similar to a stream cipher (see section A.1.2 on page 63). However a PRNG must be able to change its state by processing input values that may be unpredictable to an attacker.

A 'random value' is a sample of a random variable which has a uniform distribution over the entire set of  $n$ -bit vectors, for some  $n$ . Actually a PRNG is the heart of a cryptographic system. Often not much attention is paid to PRNGs, but they are the basis of a lot of cryptographic systems. Random numbers are in session keys, nonces, initialization vectors, public-key generation, and many other places. If the random numbers are insecure, then the entire application is insecure. Algorithms and protocols that use bad random numbers cannot cover the fact they use them. As an example: in 1995, Berkeley students broke the security of Netscape Navigator by analyzing the PRNG. In theory, true random numbers only come from truly random sources: atmospheric noise, radioactive decay, political press announcements. If a computer generates the number, another computer can reproduce the process. A sequence generator is pseudo random if (1) it looks random, (2) it is unpredictable and (3) it cannot be reliably reproduced. (1) means that it passes all statistical tests of randomness that can be found.

For a sequence to be *cryptographically secure* pseudo random (CSPR) it must not only comply to the statistical randomness, but it must also be unpredictable. This means that given the complete algorithm or hardware generating the sequence and all previous bits in the stream it must be computationally unfeasible to predict what the next bit will be. The seed to set the initial state of the generator is often called the key and without it CSPR sequences should not be compressible.

A sequence generator is *real random* if it cannot be reliably reproduced. Meaning if the sequence generator is run twice with the exact same input (as far as humanly possible), still two completely unrelated random sequences are generated.

Only the output of a sequence generator that complies to all three properties mentioned is good enough for one-time pad- or key generation and all other applications that need a truly random sequence generator.

### A.7.1. Attacks on PRNGs

Attacks on PRNGs can make a careful selection of algorithms and protocols for crypto systems irrelevant. So the PRNG is a single point of failure for many crypto systems. There are different types of attack

- **Direct Cryptanalytic Attack:** PRNG output and random output can be distinguished. This attack can be mounted to most PRNGs.
- **Input Based Attack:** the attacker is able to use control or knowledge of the PRNG inputs to distinguish between PRNG output and random output.
- **State Compromise Extension Attacks:** recovering unknown PRNG output (or distinguish from random output) from an internal state  $S$  that was once compromised. This unknown output can be output from before or after  $S$  was compromised. One can discriminate:
  - **Backtracking Attack:** use  $S$  to learn previous PRNG outputs.
  - **Permanent Compromise Attack:** all future and past  $S$  values are vulnerable to attack.
  - **Iterative Guessing Attack:** by  $S$  at time  $t$  the attacker can learn  $S$  at  $t + \delta$  by guessing the inputs collected during this span of time.
  - **Meet-in-the-Middle Attack:** combination of iterating guessing and backtracking. From  $S$  at times  $t$  and  $t + 2\delta$ ,  $S$  at time  $t + \delta$  can be recovered.

There are also examples where PRNGs are used incorrectly. For example, if it is known that a cryptographic system uses 128 bit keys, but these keys are derived from a known fixed PRNG that has an 8 bit seed. Then an attacker only needs to search  $2^8 = 256$  keys (using every possible seed used for the PRNG). And not the  $2^{128}$  possible keys according to the system.

### A.7.2. Skew Correction

Sometimes the chance of producing a 1 and the chance of producing a 0 in a random number generator (rather: random bit generator) is not the same. This means the distribution of 0's

and 1's is not uniform and hence the produced sequence cannot be stated to be random. This non-uniform distribution is called skew. Fortunately there are skew correction algorithms that can convert the sequence into one with (almost) uniform distribution.

### De-Skew by Transition Mapping

This skew correction algorithm is based on the transitions in the original sequence. Bits are read two at a time, and if there is a transition between values (01 or 10) one of them (e.g the first) is passed on as random. If there is no transition (00 or 11), the bits are discarded and the next two are read. This simple algorithm is called the Von Neumann strategy. It completely eliminates any bias towards 0 or 1 in the samples. This can be seen as follows: assume the probability of a 1 is  $P(1) = p = \frac{1}{2} + e$  and the probability of a 0 is  $P(0) = q = \frac{1}{2} - e$  where  $e$  is the eccentricity of the source. Table A.4 shows the probability of each pair. The drawback

pair	probability
00	$(\frac{1}{2} - e)^2 = \frac{1}{4} - e + e^2$
01	$(\frac{1}{2} - e) \cdot (\frac{1}{2} + e) = \frac{1}{4} - e^2$
10	$(\frac{1}{2} + e) \cdot (\frac{1}{2} - e) = \frac{1}{4} - e^2$
11	$(\frac{1}{2} + e)^2 = \frac{1}{4} + e + e^2$

Table A.4.: Von Neuman strategy probabilities

of this algorithm is that it takes an indeterminate number of input bits. The probability of a pair being discarded is:  $P(00) + P(11) = \frac{1}{2} + 2e^2$ . So to produce  $X$  output bits it is expected to need  $X/(\frac{1}{4} - e^2)$  input bits. Which means 75% of the input stream is lost, when the bit stream is already unbiased.

### De-Skew by Stream Parity

Of course the transition mapping can be extended to a parity mapping of  $N$  input bits to one bit. This mapping will not be a perfect uniform distribution, but can come as close as desired (at the cost of number of the bits that need to be sampled to generate a de-skewed sequence length). The probability that the parity will be 1 or 0 is the sum of all the possible odd or even terms with length  $N$  (where odd means an odd number of 1's). All these terms together are the binomial expansion of  $(p + q)^N$ . Where the terms can be split up in

$$\begin{aligned} \text{Terms}_1 &= \frac{1}{2} ((p + q)^N + (p - q)^N) \\ &= \frac{1}{2} (1 + (2e)^N) \end{aligned} \tag{A.9}$$

$$\begin{aligned} \text{Terms}_2 &= \frac{1}{2} ((p + q)^N - (p - q)^N) \\ &= \frac{1}{2} (1 - (2e)^N) \end{aligned} \tag{A.10}$$

with:

$\text{Terms}_1$  : Sum of all terms with an even number of 1's when  $N$  is even and an odd number of 1's when  $N$  is odd.



Terms<sub>2</sub> : Sum of all terms with an odd number of 1's when  $N$  is even and an even number of 1's when  $N$  is odd.

$p$  : =  $\frac{1}{2} + e$ , probability of a 1.

$q$  : =  $\frac{1}{2} - e$ , probability of a 0.

$e$  : Eccentricity of the source

$N$  : Number of input bits

If probabilities are wanted that are within  $d$  of  $\frac{1}{2}$  in the output, this leads to:

$$\begin{aligned} \frac{1}{2}(1 + (2e)^N) &= \frac{1}{2} + d \\ N &> \frac{\log(2d)}{\log(2e)} \end{aligned} \tag{A.11}$$

Table A.6 shows as example the number of input bits needed to de-skew a stream (skewness indicated by the probability of a 1) to a  $d$  value of 0.001.

<b>P(1)</b>	$e$	$N$
0.5	0.00	1
0.7	0.20	7
0.9	0.40	28
0.99	0.49	308

Table A.6.: Example: input bits needed to de-skew stream

### De-Skew by Compression

If a stream is reversibly compressed, the Shannon information equation states that this is only possible if the probabilities of the shorter sequences of the compressed sequence, are more uniformly distributed than the probabilities of the original. So the shorter sequences are de-skewed. However, there seems to be a subtle introduction of patterns in the output with many compression algorithms [14].

### De-Skew by FFT

As seen in table A.6, data that is highly biased can still contain useful amounts of randomness. By using the discrete Fourier Transform the randomness can be extracted, by discarding strong correlations. If enough data is processed and the correlations decline, the spectral lines will approach statistical independence. From this normally distributed data can be produced.

## A.8. Entropy

This section explains more about the amount of randomness a variable or measurement value contains. Equation A.12 on the next page was already used in section A.2 to calculate the entropy of a key and a plaintext.

### A.8.1. Bits of Information

Section A.7.2 discussed the de-skewing of sequences generated from non-uniform distributions, but did not say anything about how much information a sequence possesses. The amount of information present in a message depends on the number of values possible and the probability of each value. A message can for example be a secret key. Shannon [25] derived that the entropy of a message is

$$H = \sum_{i=1}^n (-p_i \log_2(p_i)) \quad (\text{A.12})$$

with:

- $H$  : Bits of information present in a message
- $n$  : Number of possible values
- $p_i$  : Probability of the value numbered  $i$  ( $p_i \leq 1$ )

On average an attacker would have to use half of the values to guess the message. So  $h$  bits of information, would mean  $2^{n-1}$  tries. But this is only valid if all values have equal probability (uniform distribution). If their probability is unequal then on average fewer guesses satisfy, because the attacker will search through the more probable values first.

### A.8.2. Entropy of a measurement value

When measured variables are converted to keys that are used in cryptography, it is important to know how much information a measurement value contains. Shannon's Noisy Channel Coding Theorem [4, 25] states how much information is in a measurement value. The theorem is shortly discussed here. Consider a discrete-time Gaussian channel

$$y_i = x_i + n_i \quad (\text{A.13})$$

with:

- $y_i$  : Output
- $x_i$  : Information bearing variable
- $n_i$  : Gaussian random variable (noise) with variance  $\sigma_n^2$

The input  $x_i$ 's are constrained to have power:

$$\frac{1}{N} \sum_{i=1}^N (x_i^2) \leq p \quad (\text{A.14})$$

with:

- $p$  : Power of the signal, equals:  $\sigma_s^2$  (the variance of the signal)
- $N$  : Block size

Consider an output block of size  $N$  ( $N$  uses of a single channel, note that characters in boldface are vectors):

$$\mathbf{y} = \mathbf{x} + \mathbf{n} \quad (\text{A.15})$$

For large  $N$ , by the Law of Large Numbers,

$$\frac{1}{N} \sum_{i=1}^N (n_i^2) = \frac{1}{N} \sum_{i=1}^N |y_i - x_i|^2 \leq \sigma_n^2 \quad (\text{A.16})$$

$$\frac{1}{N} |\mathbf{y} - \mathbf{x}|^2 \leq \sigma_n^2 \quad (\text{A.17})$$

$$|\mathbf{y} - \mathbf{x}| = \sqrt{N\sigma_n^2} \quad (\text{A.18})$$

This indicates that for large  $N$ ,  $\mathbf{y}$  will -with a high probability- be located in an  $N$ -dimensional sphere of radius  $\sqrt{N\sigma_n^2}$  centered about  $\mathbf{x}$ . Since  $x_i$ 's are power constrained and  $n_i$ 's and  $x_i$ 's are independent:

$$\frac{1}{N} \sum_{i=1}^N y_i^2 \leq p + \sigma_n^2 \quad (\text{A.19})$$

$$|\mathbf{y}| \leq \sqrt{N(\sigma_n^2 + p)} \quad (\text{A.20})$$

Which means  $\mathbf{y}$  is in a sphere of radius  $\sqrt{N(\sigma_n^2 + p)}$  centered around the origin. So how many  $\mathbf{x}$ 's can be transmitted to have non-overlapping  $\mathbf{y}$  spheres in the output domain? The question is how many spheres of radius  $\sqrt{N\sigma_n^2}$  fit in a sphere of radius  $\sqrt{N(\sigma_n^2 + p)}$ :

$$\frac{V_y}{V_{y|x}} = \frac{\frac{4}{N}\pi(r_y)^N}{\frac{4}{N}\pi(r_y|x)^N} \quad (\text{A.21})$$

$$= \frac{\left(\sqrt{N(\sigma_n^2 + p)}\right)^N}{\left(\sqrt{N\sigma_n^2}\right)^N} \quad (\text{A.22})$$

$$= \left(1 + \frac{p}{\sigma_n^2}\right)^{\frac{N}{2}} \quad (\text{A.23})$$

with:

$V_y$  : Volume of all probable  $\mathbf{y}$

$V_{y|x}$  : Volume of  $\mathbf{y}$  given  $\mathbf{x}$

The bits of information that can be sent across this channel in one use is ( $N = 1$ ):

$$\log_2 \left(1 + \frac{p}{\sigma_n^2}\right)^{\frac{1}{2}} = \frac{1}{2} \log_2 \left(1 + \frac{p}{\sigma_n^2}\right) \quad (\text{A.24})$$

This formula can be used to calculate the number of bits a measurement value carries about a certain object.



## B. Plotting Functions

Some functions in the report need re-writing in order to plot them. This section shows this rewriting as well as some proofs.

### B.1. Plotting $q \cdot f_W(w|s = j)$

To plot  $q \cdot f_W(w|s = j)$  against  $w/q$  for different values of  $q/\sigma_x$ ,  $q \cdot f_W(w|s = j)$  has to be rewritten. This section shows how to rewrite it for  $|w| \leq q$ .

$$\begin{aligned}
 q \cdot f_W(w|s = j) &= q \cdot \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{((vn+a_j)q-w)^2}{2\sigma_x^2}} \\
 &= \sum_{n=-\infty}^{\infty} \frac{q}{\sqrt{2\pi}\sigma_x} e^{-\frac{q^2((vn+a_j)-w/q)^2}{2\sigma_x^2}}
 \end{aligned} \tag{B.1}$$

The result can easily be plotted. Note that  $v$  is the number of values that is quantized to and  $a_j$  is defined as in equation 1.7 on page 15.

### B.2. Plotting $I(W; S)$

To plot  $I(W; S)$  against  $\sigma_x/q$ , equation 1.23 on page 20 has to be rewritten. This section shows how to rewrite it. The original equation:

$$I(W; S) = \int_{-q}^q f_W(w|s = 1) \log_2 f_W(w|s = 1) dw - \int_{-q}^q f_W(w) \log_2 f_W(w) dw$$

First the left part of the minus sign is rewritten:

ps: [...] indicates that the dots are the same, as the part in the formula that is contained between [] (other type of brackets are used in the same way). For example  $[A+(B \cdot C)]+[...] =$

$$[A + (B \cdot C)] + [A + (B \cdot C)].$$

$$\begin{aligned}
& \int_{-q}^q f_W(w|s=1) \log_2 f_W(w|s=1) \, dw \\
&= \int_{-q}^q \left[ \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{((2n+1/2)q-w)^2}{2\sigma_x^2}} \right] \log_2 [\dots] \, dw \\
&= \{z = w/q, \, dz = dw/q, \, dw = q \, dz (w = q \Rightarrow z = 1, w = -q \Rightarrow z = -1)\} \\
& \int_{-1}^1 \left[ \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{((2n+1/2)-z)^2}{2(\sigma_x/q)^2}} \right] \log_2 [\dots] \, q \, dz \\
&= \{\text{bring } q \text{ to the front, the term under the log is multiplied by } q/q\} \\
&= \int_{-1}^1 \left[ \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n+1/2)-z)^2}{2(\sigma_x/q)^2}} \right] \log_2 \frac{1}{q} [\dots] \, dz \\
&= \int_{-1}^1 [\dots] (\log_2 [\dots] - \log_2 q) \, dz \\
&= \int_{-1}^1 [\dots] \log_2 [\dots] - [\dots] \log_2 q \, dz \\
&= \int_{-1}^1 \left[ \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n+1/2)-z)^2}{2(\sigma_x/q)^2}} \right] \log_2 [\dots] \, dz - \log_2 q \int_{-1}^1 [\dots] \, dz \quad (\text{B.2})
\end{aligned}$$

The first term, left of the minus sign can be plotted, the right however still contains a  $q$  in it. Next, the right part is rewritten:

$$\begin{aligned}
& \int_{-q}^q f_W(w) \log_2 f_W(w) dw \\
&= \{f_W(w) = f_W(w|s=0)P(s=0) + f_W(w|s=1)P(s=1), z = w/q\} \\
& \int_{-1}^1 \left[ \frac{1}{2} \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n-1/2)-z)^2}{2(\sigma_x/q)^2}} + \frac{1}{2} \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n+1/2)-z)^2}{2(\sigma_x/q)^2}} \right] \log_2 \left( \frac{1}{q} [\dots] \right) dz \\
&= \int_{-1}^1 [\dots] \cdot (\log_2 [\dots] - \log_2 q) dz \\
&= \int_{-1}^1 [\dots] \log_2 [\dots] dz \\
&\quad - \log_2 q \left[ \frac{1}{2} \int_{-1}^1 \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n-1/2)-z)^2}{2(\sigma_x/q)^2}} dz + \frac{1}{2} \int_{-1}^1 \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n+1/2)-z)^2}{2(\sigma_x/q)^2}} dz \right] \\
&= \{\text{see proof of equation B.5}\} \\
& \int_{-1}^1 [\dots] \log_2 [\dots] dz - \log_2 q \int_{-1}^1 \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n+1/2)-z)^2}{2(\sigma_x/q)^2}} dz \tag{B.3}
\end{aligned}$$

The first term, left of the minus sign can be plotted. The second term, right of the minus sign is the same as the right term from equation B.2. Since the total equation is B.2-B.3 the following result is obtained:

$$\begin{aligned}
& \int_{-1}^1 \left[ \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n+1/2)-z)^2}{2(\sigma_x/q)^2}} \right] \log_2 [\dots] dz \\
&\quad - \int_{-1}^1 \left\{ \frac{1}{2} \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n-1/2)-z)^2}{2(\sigma_x/q)^2}} + [\dots] \right\} \log_2 \{ \dots \} dz \tag{B.4}
\end{aligned}$$

This result can be plotted against  $\sigma_x/q$  (see the plot in figure 1.11 on page 24).

**The prove that:**

$$\frac{1}{2} \int_{-1}^1 \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n-1/2)-z)^2}{2(\sigma_x/q)^2}} dz = \frac{1}{2} \int_{-1}^1 \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n+1/2)-z)^2}{2(\sigma_x/q)^2}} dz \tag{B.5}$$

$$\begin{aligned}
& \frac{1}{2} \int_{-1}^1 \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n-1/2)-z)^2}{2(\sigma_x/q)^2}} dz \\
&= \{m = -n\} \\
& \frac{1}{2} \int_{-1}^1 \sum_{m=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((-2m-1/2)-z)^2}{2(\sigma_x/q)^2}} dz \\
&= \{((-2m-1/2)-z)^2 = -(2m+1/2-z)^2 = (2m+1/2+z)^2\} \\
& \frac{1}{2} \int_{-1}^1 \sum_{m=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2m+1/2)+z)^2}{2(\sigma_x/q)^2}} dz \\
&= \{y = -z, dz = -dy\} \\
& -\frac{1}{2} \int_1^{-1} \sum_{m=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2m+1/2)-y)^2}{2(\sigma_x/q)^2}} dy \\
&= \frac{1}{2} \int_{-1}^1 \sum_{m=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2m+1/2)-y)^2}{2(\sigma_x/q)^2}} dy \\
&= \{z = y, n = m\} \\
& \frac{1}{2} \int_{-1}^1 \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_x/q)} e^{-\frac{((2n+1/2)-y)^2}{2(\sigma_x/q)^2}} dz
\end{aligned} \tag{B.6}$$



## C. AKI Manual

This manual describes step by step how to operate AKI (v3). AKI stands for **APUF Key** extraction and **I**dentification system. Basically AKI has three modes of operation

1. Identifying DL701s
2. Key extraction based on threshold
3. Key extraction based on quantization and shifting measurement values

The three main sections each describe one mode of operation and two sections are added to describe how to install and start the system.

**General note:** there are some tools included with AKI that were used during development of the system. Some of these tools might contain hard coded directories and will hence not function on a different system than the original. However, they are still included since they might be handy for future development and can be easily adjusted.

### C.1. PCI-GPIB Card Installation

This section shortly describes how to install the PCI-GPIB interface card. This card is used for communication between a computer and a GPIB compatible device - such as the Hewlett Packard 8753D Network Analyzer. Software version used during the experiments: NI-488.2 For Windows v2.1.

- Insert NATIONAL INSTRUMENTS NI-488.2 FOR WINDOWS CD. It automatically starts OR start it by clicking **Start** — **Run** — **type:** “**E:\AutoRun.exe**”, where E should be your CD-Drive letter.
- Choose “**Getting Started Documentation**”.
- Choose “**Getting started Cards**”. Follow the directions in this document. During installation follow these hints
  - **Custom** — **Next**
  - When asked which card, choose **PCI-GPIB card** (driver will be added).
  - Click **Shutdown & Finish** (computer will shutdown).
  - Insert GPIB card in free PCI slot. (Connect cables as well).
  - Start computer again
  - Two windows will appear “Found new hardware” & “NI-488.2 Getting Started Wizard”. First continue with the “Found new hardware” window
    - \* Make sure the NI-488.2 CD is still inserted. Click **Next** — **Next**

- Then continue with the “NI-488.2 Getting Started Wizard” window
  - \* Click **Verify your hardware and software installation**
  - \* If everything went OK: “The NI-488.2 Troubleshooting Wizard successfully completed” — Click **OK**
  - \* Click **Exit**
  - \* Click **Communicate with your instrument** — **Next**
  - \* Follow the directions. Note that you cannot right-click the GPIB Interface Number as stated. Instead, Click **Scan For Instruments**, in the upper toolbar.
  - \* Two instruments will be detected when communicating with the Hewlett Packard 8753D Network Analyzer: Instrument0 (Primary Address 16), Instrument1 (Primary Address 17). Instrument1 is the display in the network analyzer and does not respond to \*IDN? commands.
  - \* Response of Instrument1 to \*IDN?: HEWLETT PACKARD,8753D,0,5.24

- The PCI-GPIB card is now installed.

## C.2. Starting the System

### C.2.1. Starting the Network Analyzer

Before the analyzer can be used correctly with AKI or enrollment tools, some settings have to be changed. It is possible to store the settings in the analyzer for easy recalling. Make sure to check the following important settings

Setting	Comment
Measuring mode	Should always be $S_{21}$ . This means the transfer function of channel 1 (output) to channel 2 (input) is measured (usually selecting channel 2 also selects this measuring mode)
Start	Start frequency of sweep
Stop	Stop frequency of sweep
Sweep Type Menu	Should be ‘linear frequency’
Sweep Time	Should always be the same as during enrollment

Especially check these settings when the Quantized Key Extraction mode has been used, since that mode of operation changes settings automatically. In future, automatic changes to correct settings could be implemented for the other modes of operation as well. For more information how to operate the analyzer, please consult its manual.

### C.2.2. Preparing the Analyzer

When the network analyzer (NA) is switched on it always goes into a default state with error correction turned off. The response to the GPIB ‘OUTPCAL’ command returns the default error correction (calibration) coefficients (in this particular system:  $-7.509766000000000E-01, 0.077820000000000E-01$ ). These values are only changed when a calibration is performed. Since the the *actual* values are unimportant for the test system that is built (only the *variation*

of the values), calibrations do not have to be performed. When in future such systems would be used to read out APUFs, these systems of course would all need to be calibrated to give the same response.

### C.2.3. Sweep Time

The NA can determine the sweep time automatically. Table C.2 shows some settings. Because sometimes the auto sweep time does not result in a good signal, the sweep time has to be set manually. The longer the sweep time, the better the signal throughput gets. One very strange setting is the one from line 2 in the table; the NA sometimes chooses different sweep times! . The **manual** column indicates recommended times to set. Above the maximum in

Frequency range [MHz]	Number of points	Time [ms]	
		auto	manual
2-8.2	1601	800	800
2-8.2	51	12.5/25	800
4.7-5.2	1601	800	800-1600
4.7-5.2	101	50	50-1600
4.7-5.2	51	25	25-1600

Table C.2.: Measurement times.

this column, barely differences can be seen in the output.

### Number of Points

This setting determines how many points in the frequency range the NA measures and has possible values: 3, 11, 26, 51, 101, 201, 401, 801 or 1601. Note that the output of the NA consists of a complex number (to indicate the amplitude and phase change). This means by measuring for example 51 points, actually 102 components are obtained that contain information about the object.

### Frequency Range

The frequency points to measure can be set to be from a range or from a list of segments that is entered in the NA. The segment list is handy when several not evenly distributed frequency points need to be measured. Instead of measuring each frequency point separately the list can be entered as segments consisting of only one point. Note that the list can contain a maximum of 30 segments.

### C.2.4. Plotting and Conversion of the Network Analyzer Values

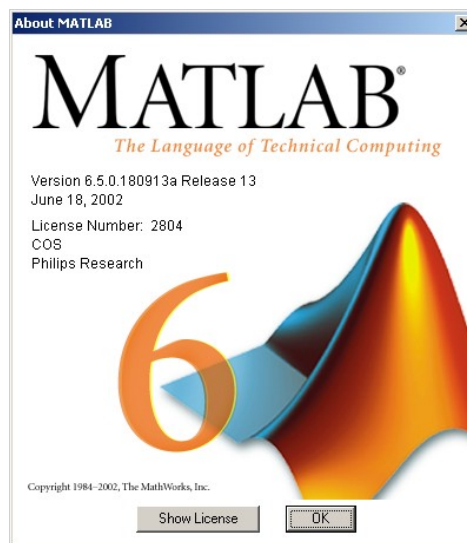
The network analyzer outputs the voltages measured [V] (a real and an imaginary part). The analyzer is able to show these values in different ways. In order to show the values the same way on a computer a conversion has to be made. In order to plot the data, the x-axis can be constructed with the Matlab command:  $f_{start} : \frac{f_{stop} - f_{start}}{Nr.Points} : f_{stop}$

Setting Name	Matlab conversion	Mathematical
'LOG MAG'	db( data )	$10 \cdot \log \left( \frac{(\sqrt{\text{Re}(\text{data})^2 + \text{Im}(\text{data})^2})^2}{R} \right)$ where $R = 1$

Table C.3.: Conversion

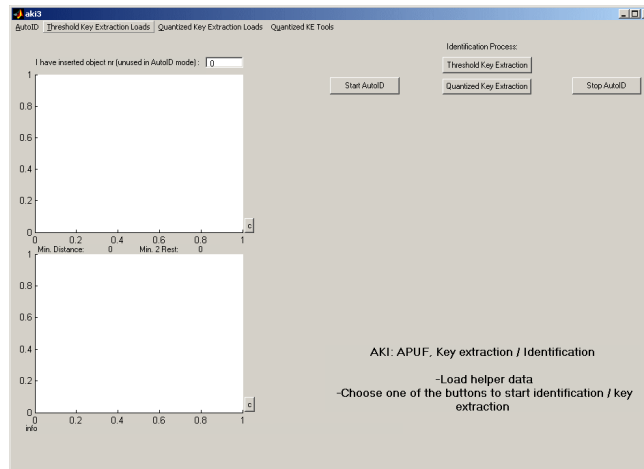
### C.2.5. Starting AKI

AKI runs under Matlab version



Older versions might work as well, but it is known that AKI does not run under Matlab Version 6.1.0.450 (R12.1).

- Find the **DefaultDir.txt** file in the AKI program dir and open it in a text editor
- Check the paths listed in it are correct and existing, an example file **DefaultDir-Format.txt** is included in case the file got corrupted. The first entry on a row is the description of the information in that row, the second entry is the value
- Start Matlab
- Change the path in the **current directory field** to the path where AKI is located
- Type **aki3** in the Matlab command window
- AKI is started and loads some images from a database. The main screen is displayed



After starting the analyzer and AKI (or the other way around) the system is ready for operation.

### C.3. Auto Identification

Auto identification mode of operation identifies DL701s automatically as they are inserted into the reader. The display will show a default screen which informs the user to insert an object and when an object is inserted the display will show information about the identification. When objects are removed, the display will return to the default screen again.

#### C.3.1. Enrollment for Auto Identification

Enrollment for auto identification is not implemented yet from within AKI. Enrollment is done with a separate set of tools. To perform an enrollment do the following

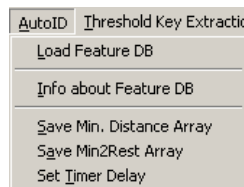
- Enter the preferred measurement settings on the analyzer
- Start `..\C\Enroll\Enroll.exe` in a dos window
- Enter the wanted settings when asked by the program
- Insert the objects as indicated on the screen. Each measurement will be saved automatically in `..\C\Enroll\Measurements`
- Start `..\Tools\createArrayf` from Matlab to combine all the individual measurements into one big measurement file
- Start `..\Tools\preparePCADData` and select the measurement file to calculate the feature values based on principle component analysis
- The data needed for auto identification has now been created

Some other tools that can be used to display information

Filename	Goal
readfile	Reads the measurement data from one individual measurement file. The measurement can then for example easily be plotted as it would be shown on the network analyzer display
plotCoeffs	Plots the principle components coefficients
showEntropy	Show the entropy in the feature value components
showComponents	Show the principle components (that are the new basis)
showEntropyMeas	Show the entropy in the measurement values

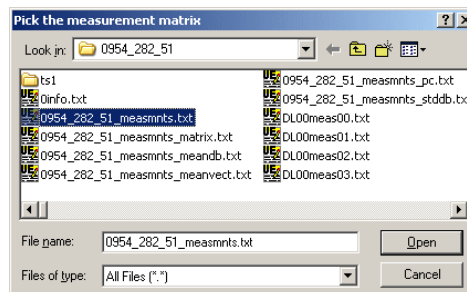
### C.3.2. Auto Identification

- Click **AutoID** from the menu bar

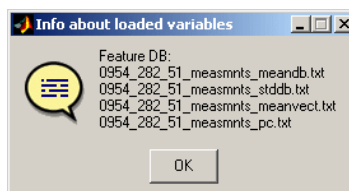


and select **Load Feature DB**

- Select the file that contains the measurements



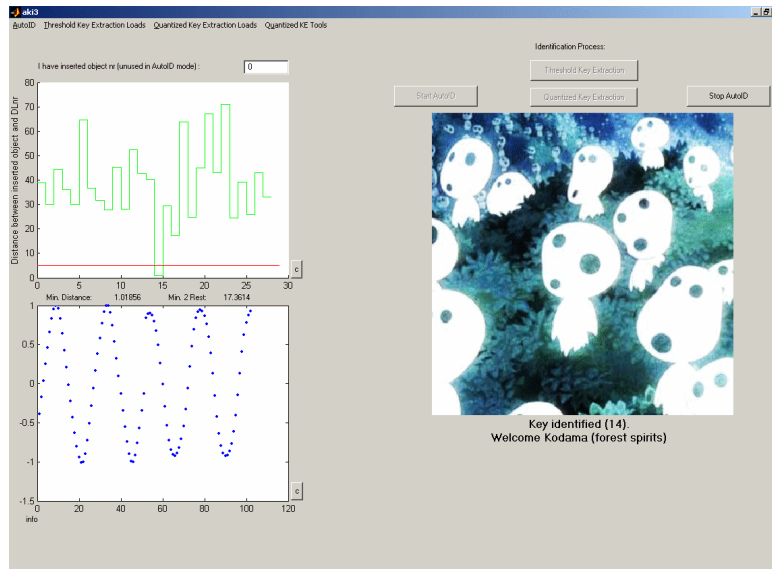
- If the files loaded correctly, an information message is displayed



Click **OK**. This message which contains information about the loaded variables can always be shown by selecting **AutoID | Info about Feature DB**

- To start the auto identification click the **Start AutoID** button in the main screen

- The system is now ready to identify DL701s. Insert objects into the reader to auto identify them. Objects not in the enrollment database will be recognized as unidentified. Below the screen when an object is identified



- To stop auto identification, click the **Stop AutoID** button in the main screen
- To fully stop this mode of operation, an object (enrolled or not) must be inserted into the reader (because Matlab makes use of an external program to perform the measurements and this program does not return control to Matlab before it has performed a valid measurement)

Note: NEVER leave an object sitting in the reader. After some time the program will crash because of a timer problem.

To save the distances measured between the measured data and the database data, click **AutoID** and then **Save Min.Distance Array** and/or **Save Min2Rest Array**. Note that for one measurement these distances are also shown in the main screen in the upper graph.

## C.4. Threshold Key Extraction

The threshold key extraction mode of operation extracts keys from DL701s based on the entered ID by the user. The display will show a default screen which informs the user to insert an object. When the key has been extracted the display will show information about the extracted key.

### C.4.1. Enrollment

Enrollment for threshold key extraction is not implemented yet from within AKI. Enrollment is done with a separate set of tools. To perform an enrollment do the following

- Follow the steps in section C.3.1 until (and including) the `..\Tools\createArrayf` step

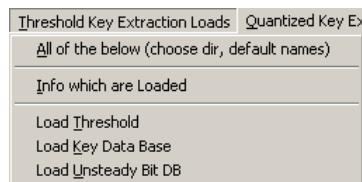
- Start **prepareData** to create test and train sets from the measurements
- Start **generateThresholdAndKeysDB** to create the threshold information and other helper data
- The data necessary for threshold key extraction has now been created

Some other tools that can be used to display information

Filename	Goal
bitstringFromData	Visualize binary keys and calculate the effective number of bits in the keys by calculating the fractional Hamming distance
FHDcalc and PlotFHD	Can be used for the same as above. First the keys should be loaded, then use $FHD = FHDcalc(keys)$ to calculate the fractional Hamming distances and $PlotFHD(FHD)$ to plot them

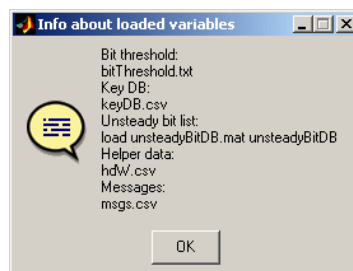
#### C.4.2. Key Extraction

- Click **Threshold Key Extraction Loads** from the menu bar



and select **All of the below**

- Select the file that contains the measurements
- If the files loaded correctly, an information message is displayed

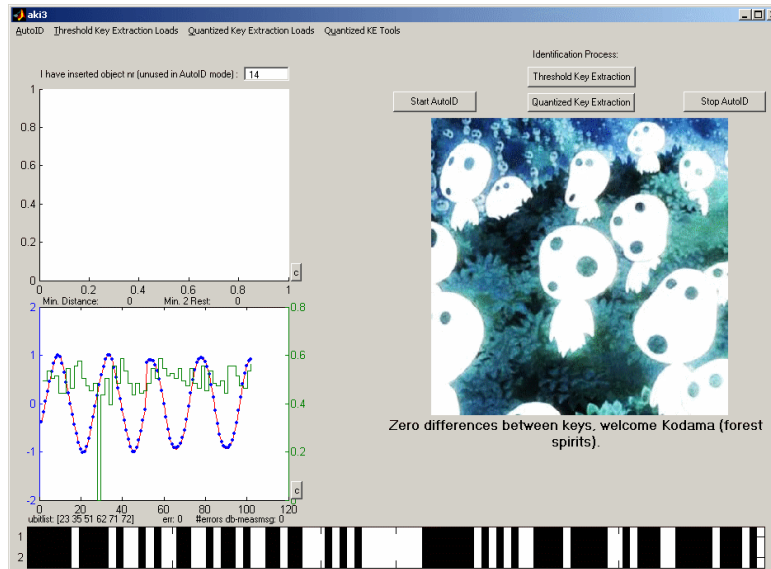


Click **OK**. This message which contains information about the loaded variables can always be shown by selecting **Threshold Key Extraction Loads | Info which are loaded**

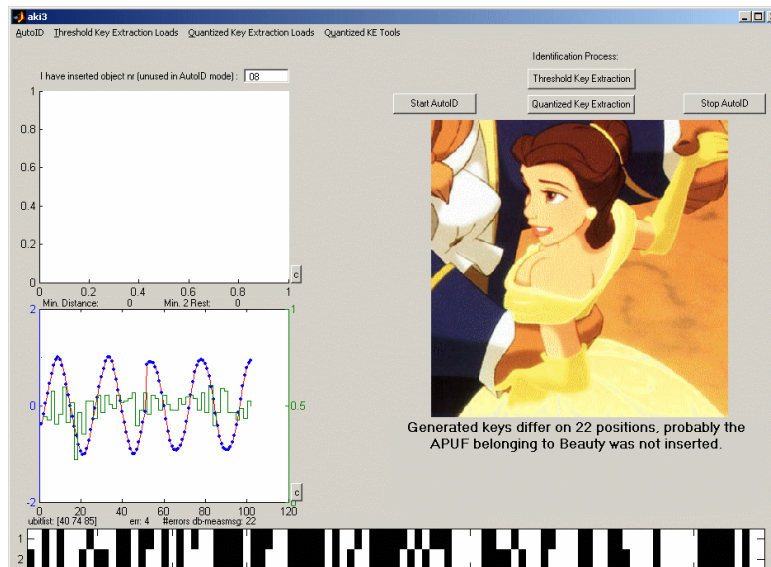
- The system is now ready to extract keys from DL701s.



- To extract a key from an inserted object, fill in the **I have inserted object nr.** field and click the **Threshold Key Extraction** button in the main screen. When key extraction has been performed the following screen is shown when a correct key is extracted



When an incorrect key is extracted

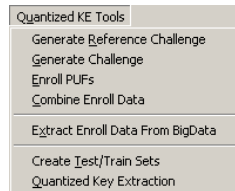


## C.5. Quantized Key Extraction

The analyzer is controlled by AKI in this mode of operation. The only setting that should be the same as during enrollment is Sweep Time. The rest is set automatically by AKI.

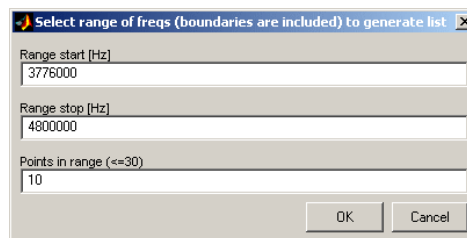
### C.5.1. Enrollment

- Click **Quantized KE Tools** from the menu bar



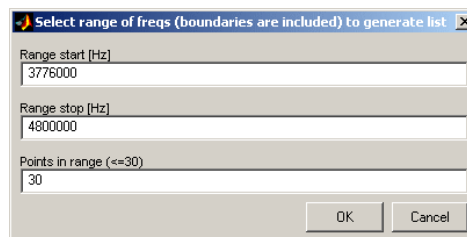
and select **Generate Reference Challenge**

- Enter the frequency range for which the reference challenge should be created



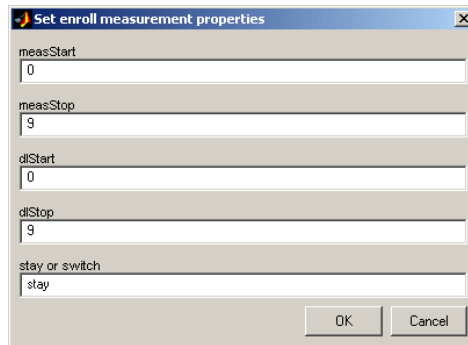
Click **OK**

- The reference challenge can be saved
- Click **Quantized KE Tools | Generate Challenge**
- Enter the frequency range from which the random challenge should be created



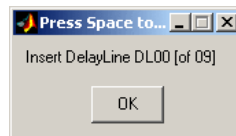
Click **OK**

- The challenge can be saved
- Click **Quantized KE Tools | Enroll PUFs**
- Enter the details how to perform the enrollment



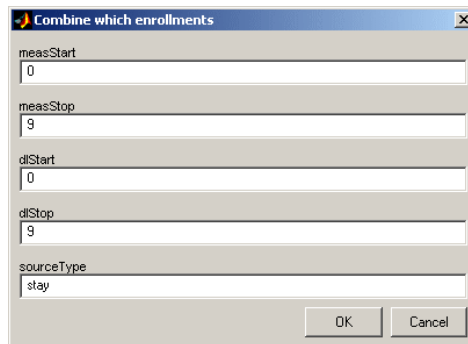
Click **OK**

- The system now starts the enroll procedure. Insert the PUFs as indicated on the screen



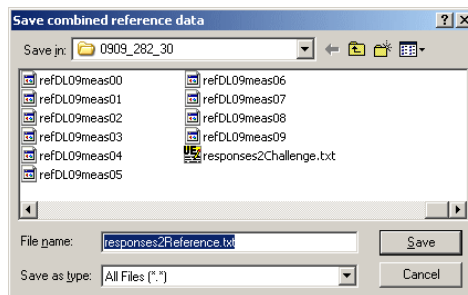
The measurements are automatically saved one by one after each measurement.

- Click **Quantized KE Tools | Combine Enroll Data**
- Enter the details which measurements to combine

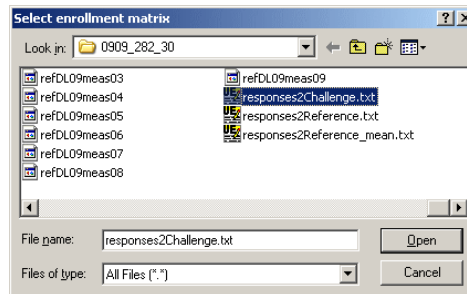


Click **OK**

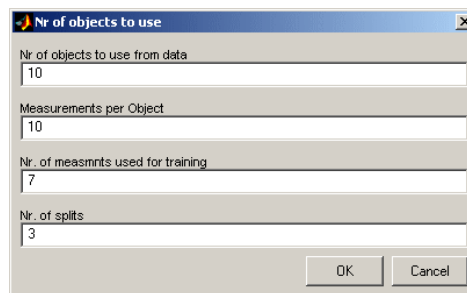
- The measurement files are combined into one file. Combined reference data and combined measurement data are saved



- Click **Quantized KE Tools | Create Test/Train Sets**
- Select the text file that contains all the responses of the objects to the challenge

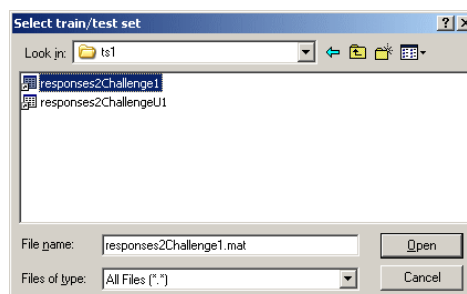


- Enter the details which measurements to include in the test train sets.

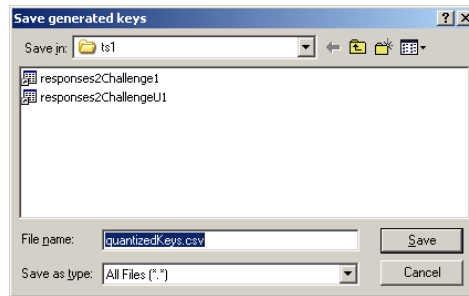


Click **OK**

- Click **Quantized KE Tools | Quantized Key Extraction**
- Select the train/test set to enroll



- The secret keys are generated and saved immediately



- The quantization factor needs to be chosen. In the Matlab command line the choice can be entered

```
total entropy: 2.967474e+002
mean entropy per component: 4.945791e+000
MkeyValue [ 2^floor(entropy) ]: [16 16 16 32 16 32 32 32 32 32]
New entropy [ 2log( product(MkeyValue) ) ]: 263
sizekeyDB: 10 60
Choose factor (q=factor*sigmaNoise, default=6, 0=end):
```

- AKI shows the results after generating the helper data and using it on the measurement data (see figure C.1 on the next page and C.2 on page 93). The Matlab command line also provides information about the FAR and FRR for the chosen q

```
Choose factor (q=factor*sigmaNoise, default=6, 0=end): 3
sizeFRR:
ans =
    10    10

Mtrain: ?FRRtrain:
ans =
     1     5
     2     7
     3     5
     4     6
     5     7
     6     6
     7     6
     8     4
     9     7
    10     6

FRRtest:
ans =
     1     3
     2     3
     3     3
     4     2
     5     3
     6     3
     7     2
     8     3
     9     3
    10     3

FAR: 0

Choose factor (q=factor*sigmaNoise, default=6, 0=end): 8
sizeFRR:
ans =
    10    10

Mtrain: ?FRRtrain:
ans =
Empty matrix: 0-by-2

FRRtest:
ans =
Empty matrix: 0-by-2

FAR: 0
```

A factor of 8 was chosen during the experiments

- Enter 0 at the Matlab command line when the results are satisfactory and the quantization is as wanted
- The quantization is saved

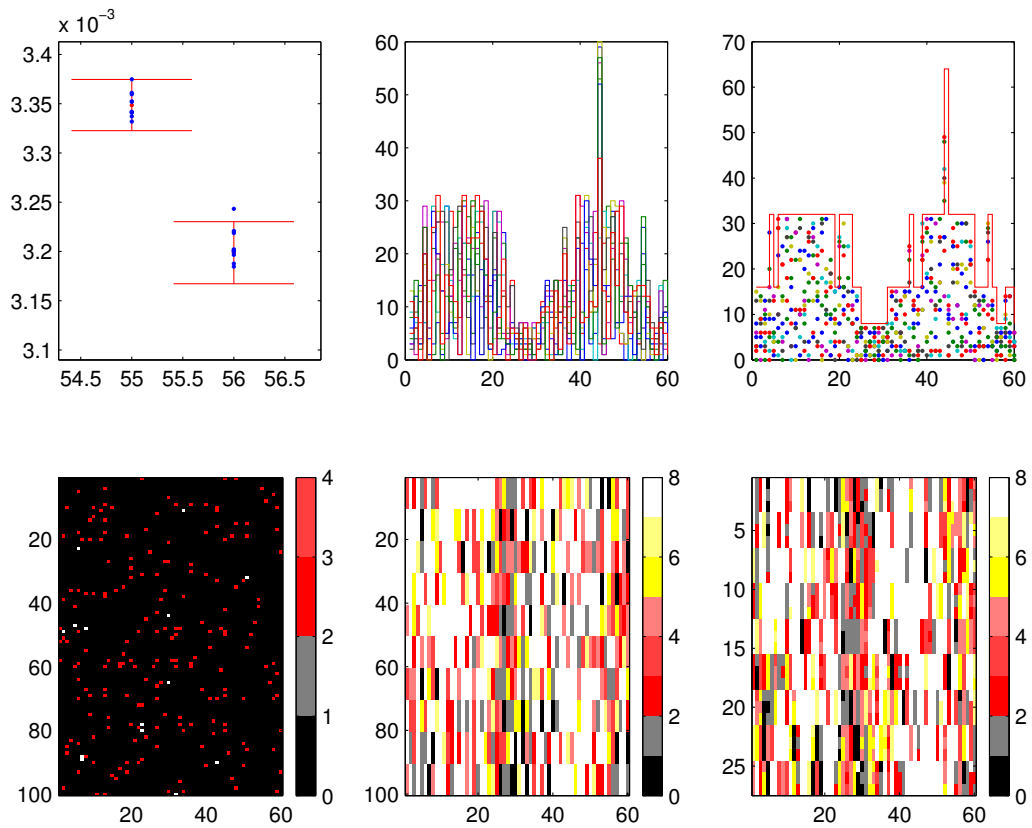


Figure C.1.: Helper figures to determine  $q$  (here  $q = 3$ ).

Left Upper: Quantization vs measurement spread (blue points should fall within the red lines)

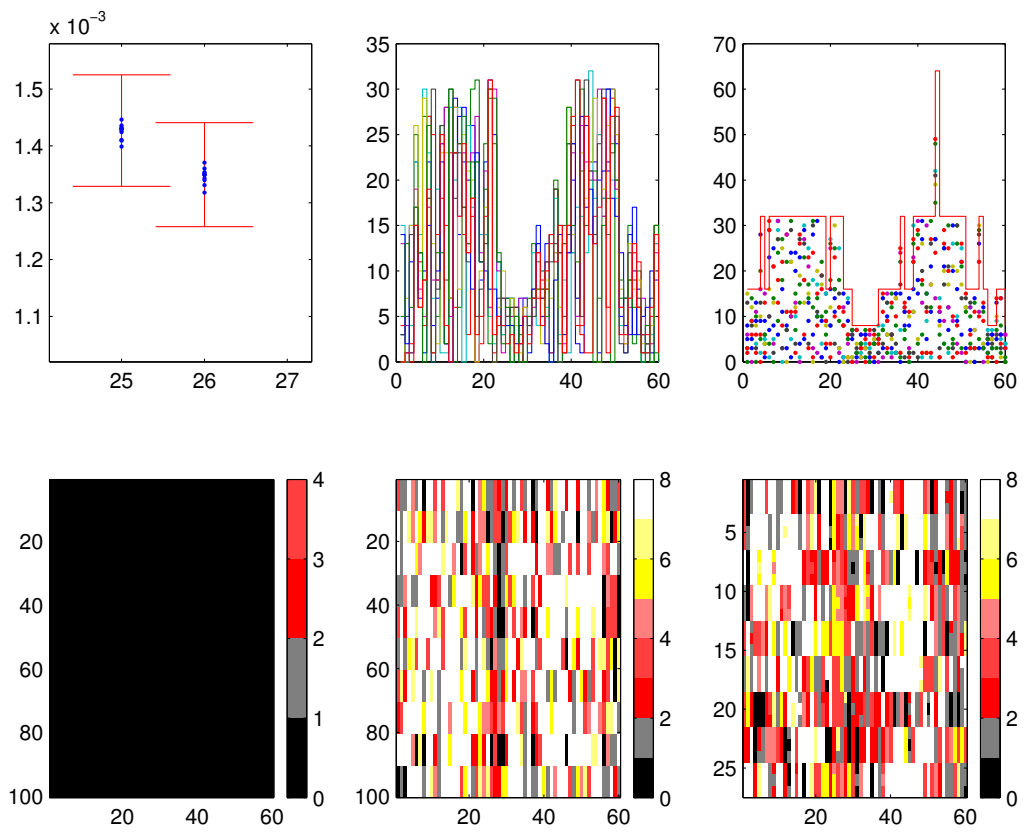
Middle Upper: Keys generated directly from the measurements without adding confusion

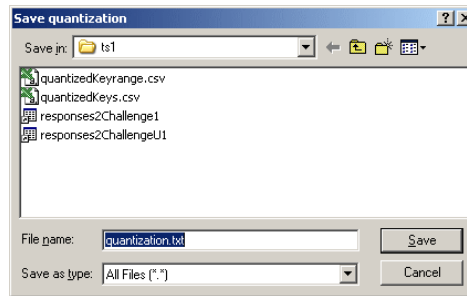
Right Upper: Keys with added confusion

Left Lower: Difference between the keys in the database and generated keys

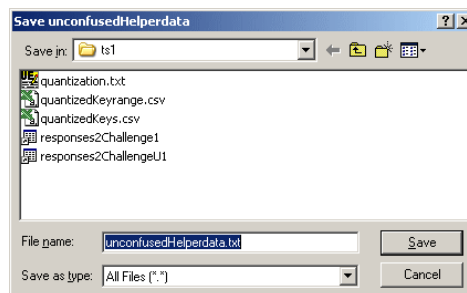
Middle Lower: Difference when keys are generated from a measurement signal that is zero (to check helper data only can be used to generate the right keys)

Right Lower: An intruder key is chosen and used to generate keys with all helper data of others and the result is compared with the key belonging to the helper data (to check FAR)

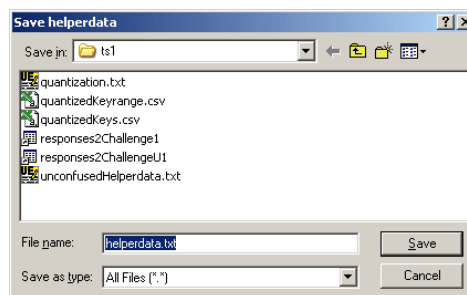
Figure C.2.: Helper figures to determine  $q$  (here  $q = 8$ ).



- The unconfused helper data is saved



- The helper data is saved



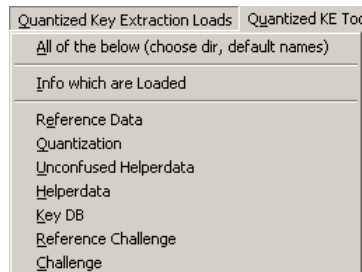
All data necessary for Quantized Key Extraction has now been generated and saved.  
Some other tools that can be used to display information

Filename	Goal
Plot_DecUnconfusedKeys.m	Visualize multi valued keys
PlotFHDnonBitKey	Plot the fractional Hamming distance between non binary keys
unconfusedKeys2Bin	Can be used to convert the multi valued keys to binary keys, after this the tools from threshold key extraction can be used to to calculate the effective number of bits



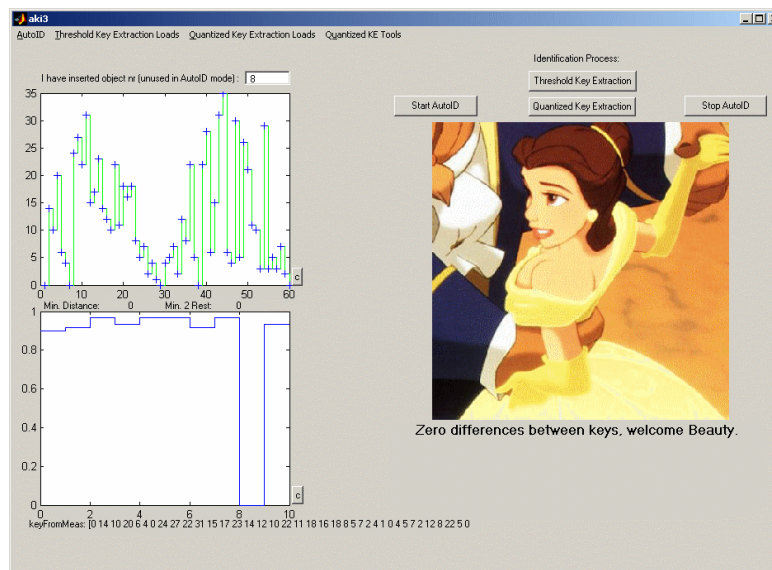
## C.5.2. Key Extraction

- Click **Quantized Key Extraction Loads** from the menu bar

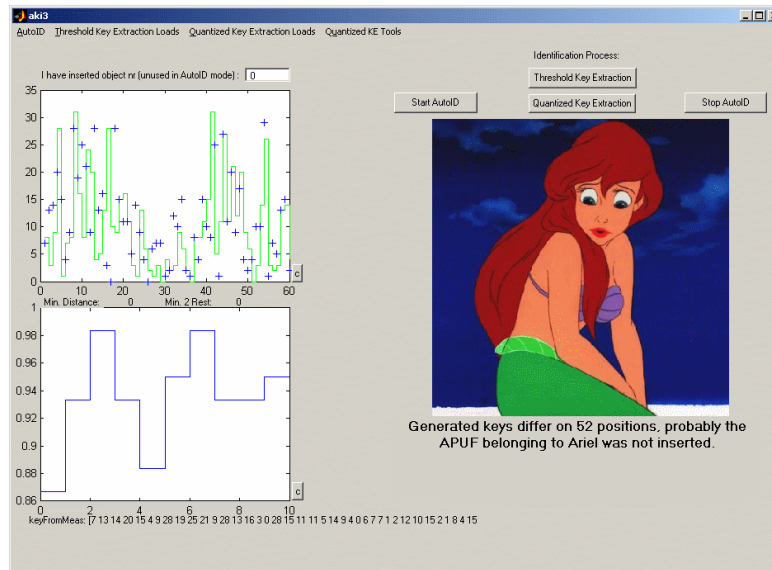


and select **All of the below**

- Select the file that contains the measurements
- If the files loaded correctly, an information message is displayed Click **OK**. This message which contains information about the loaded variables can always be shown by selecting **Quantized Key Extraction Loads | Info which are loaded**
- The system is now ready to extract keys from DL701s.
- To extract a key from an inserted object, fill in the **I have inserted object nr.** field and click the **Quantized Key Extraction** button in the main screen. When key extraction has been performed the following screen is shown when a correct key is extracted



When an incorrect key is extracted



Although this manual is absolutely not complete, hopefully it has helped in explaining how to use AKI. Many of the mentioned tools and AKI itself contain much comment in the code which will help even more in understanding AKI. For any further questions or information about the AKI system, please email to:

**vrij@gmx.net**  
*Serge Vrijaldenhoven*

last page of manual