Physical aspects of digital security
(2IC35)

Lecture notes

B. Škorić
TU Eindhoven
Dept. of Mathematics and Computer Science
Security Group (SEC)
Metaforum 6.059
http://security1.win.tue.nl/~bskoric/physsec/

v1.5, August 2014
Contents

1 Introduction .................................................................................................................... 7
   1.1 Aim of this course ..................................................................................................... 7
   1.2 Physics .................................................................................................................... 7
   1.3 Which topics are covered ......................................................................................... 7
   1.4 Required starting knowledge ................................................................................... 8

2 Information theory ........................................................................................................... 9
   2.1 Probabilities ............................................................................................................ 9
      2.1.1 Random variables .............................................................................................. 9
      2.1.2 Expectation value ............................................................................................ 11
      2.1.3 Moments .......................................................................................................... 11
      2.1.4 Statistical distance ........................................................................................... 11
      2.1.5 Joint distributions; independence; covariance .................................................. 12
      2.1.6 Marginal distributions ...................................................................................... 12
      2.1.7 Conditional probabilities ............................................................................... 12
   2.2 Shannon entropy ..................................................................................................... 13
      2.2.1 Measuring the amount of information .............................................................. 13
      2.2.2 Differential Entropy .......................................................................................... 14
      2.2.3 History of the entropy concept ........................................................................ 15
   2.3 Kullback-Leibler distance ......................................................................................... 15
   2.4 Conditional entropy and mutual information ......................................................... 15
   2.5 Rényi entropy ......................................................................................................... 17
   2.6 Error correcting codes ........................................................................................... 18
      2.6.1 Linear binary codes ........................................................................................... 18
      2.6.2 Generator matrix and parity check matrix ....................................................... 19
      2.6.3 Syndrome decoding .......................................................................................... 20
      2.6.4 A rule of thumb ............................................................................................... 20
      2.6.5 Shannon bound ............................................................................................... 21
      2.6.6 Additive Gaussian noise ................................................................................. 22

3 Random number generation ............................................................................................ 23
   3.1 Truly random numbers ............................................................................................ 23
   3.2 Examples of randomness sources ............................................................................ 23
   3.3 Making it uniform ................................................................................................... 24
      3.3.1 Known continuous source ............................................................................... 24
      3.3.2 Known discrete source; difference with the continuum case ............................. 25
      3.3.3 Known discrete source; some simple examples ............................................... 26
      3.3.4 More generic algorithms .................................................................................. 28
   3.4 Privacy amplification using public randomness ....................................................... 29
      3.4.1 Universal Hash Functions and the Leftover Hash Lemma ............................... 29
      3.4.2 Examples of (almost) universal hash functions ............................................... 33
4 Key agreement from correlated randomness 35
  4.1 Exploiting noise .............................................. 35
  4.2 The binary symmetric channel ................................ 36
  4.3 Scenario 1: Noisy broadcast by Alice ....................... 36
    4.3.1 Noisy broadcast channel (without return channel) .... 36
    4.3.2 Noisy broadcast channel plus public return channel ... 38
  4.4 Scenario 2: the satellite scenario .......................... 39

5 Physical Unclonable Functions (PUFs) 43
  5.1 An Introduction to PUFs ...................................... 43
    5.1.1 History and concepts .................................. 43
    5.1.2 A short overview of PUF realisations .................. 45
    5.1.3 Applications and use models .............................. 46
    5.1.4 Comparison with biometrics .............................. 46
    5.1.5 Current directions of PUF research ...................... 47
  5.2 A very quick introduction to Fuzzy Extractors ................ 47
  5.3 Examples of PUFs .............................................. 48
    5.3.1 Optical PUFs ............................................ 48
    5.3.2 Coating PUFs ............................................ 50
    5.3.3 SRAM PUFs .............................................. 51
    5.3.4 Delay based PUFs ........................................ 52
    5.3.5 Comparison of PUF types ................................ 53
  5.4 Detailed list of PUF applications ............................. 54
    5.4.1 Anti-counterfeiting with bare PUFs ...................... 54
    5.4.2 Remote authentication of a bare PUF .................... 55
    5.4.3 Remote authentication of a bare PUF, plus authenticated channel .................. 56
    5.4.4 Read-proof, tamper evident storage of keys ............. 57
    5.4.5 Generic anti-counterfeiting using POK'ed RFID tags ..... 57
    5.4.6 A POK without Public Key Infrastructure ............... 59
    5.4.7 Certified measurements .................................. 60
    5.4.8 Controlled PUFs ......................................... 60
    5.4.9 Reconfigurable POKs for Trusted Computing ................ 61
    5.4.10 Software-to-hardware binding ........................... 61
    5.4.11 Remote authentication of Public PUFs ................... 62
  5.5 Entropy of PUFs .............................................. 63

6 Fuzzy Extractors and Secure Sketches 67
  6.1 Definitions .................................................... 67
  6.2 When to use the FE and SS primitives ....................... 69
  6.3 The Code Offset Method ....................................... 69
    6.3.1 The original Code Offset Method ....................... 69
    6.3.2 The Syndrome-Only Code Offset Method .................. 70
  6.4 Fuzzy extractor based on partitions ......................... 71
  6.5 A fuzzy extractor based on universal hash functions ........ 73
  6.6 A helper data scheme for coating PUFs ..................... 74
  6.7 A helper data scheme for optical PUFs ..................... 75
  6.8 A helper data scheme for SRAM PUFs ........................ 76

7 Distance bounding 77
  7.1 Why distance bounding? ...................................... 77
  7.2 General design principles .................................... 78
  7.3 The Brands-Chaum protocol .................................. 79
  7.4 The “Swiss knife” protocol .................................. 81
  7.5 Analog challenge-response processing ....................... 82
8 Quantum physics

8.1 The physics of very small actions ........................................... 85
8.2 Linear algebra refresher ...................................................... 86
8.3 Dirac notation ................................................................. 87
8.4 Electron spin ................................................................. 88
8.5 Polarization of light .......................................................... 90
  8.5.1 Classical polarization .................................................... 90
  8.5.2 Polarization of a single photon ....................................... 90
8.6 Qubits .............................................................................. 92
8.7 Combining multiple quantum systems ..................................... 93
  8.7.1 Tensor products ............................................................ 93
  8.7.2 Measurement on a subsystem ......................................... 94
8.8 How to exploit all this ....................................................... 95

9 Quantum Key Distribution ....................................................... 97

9.1 The BB84 protocol .............................................................. 97
  9.1.1 Attack model and physical context .................................... 97
  9.1.2 The protocol ............................................................... 98
  9.1.3 The security of BB84 .................................................... 98
  9.1.4 Remarks about quantum key distribution ......................... 100

10 Quantum computers ............................................................ 101

10.1 Operations on qubits: quantum gates ...................................... 101
10.2 Shor’s algorithm for factoring .............................................. 103
  10.2.1 Preliminaries ............................................................ 103
  10.2.2 The classical part of the algorithm ................................. 104
  10.2.3 The quantum part of the Shor algorithm ......................... 104
  10.2.4 Remarks about the Shor algorithm ............................... 106
10.3 Shor’s algorithm for discrete logarithms ................................ 107
  10.3.1 Intuition ................................................................. 107
  10.3.2 The algorithm ......................................................... 107
Chapter 1

Introduction

1.1 Aim of this course

The security of ciphers and protocols is not a purely mathematical affair. Electronic devices such as computers and smartcards are part of a physical world; indeed, computation itself is a physical process. On the one hand, this limits the performance of processors and allows attackers to target not the mathematics of a security system but the physical implementation. So-called side channel attacks such as power, emission and timing analysis are well known, potent attacks. (They are treated in other courses of the Kerckhoffs curriculum.) On the other hand, the existence of a physical world outside (and inside!) digital devices is a great resource. One of the main prerequisites for security algorithms is to have a reliable supply of random bits. Unpredictability lies at the very heart of cryptography. It is important to remember that all true randomness is physical in origin. The only way to get truly random bits is to do measurements on something noisy.

This course is about the interface between digital security and the physical world. We cover a number of ways in which digital devices can exploit the physical world for the improvement of security.

1.2 Physics

This is not a physics course. When we discuss a device interacting with a physical system, we attempt to abstract the physics away to a point where only the main statistical properties of the interaction are left over. The emphasis will always be on the algorithms that process the physical inputs.

That having been said, it is impossible to completely ignore physics. Where necessary we include a summary of the most relevant rules and equations. In particular, we will inevitably have to dive into quantum physics; otherwise it would not be possible to explain beautiful subjects such as quantum key distribution and quantum computers. This will be done in an as painless way as possible.

1.3 Which topics are covered

- True random number generation.
- Key agreement from correlated randomness.
- Physical Unclonable Functions (PUFs).
- Secure Sketches and Fuzzy Extractors.
1.4 Required starting knowledge

This course can in principle be done without prior knowledge from other courses. However, several chapters make heavy use of information theory, and random variables in general. Some affinity with stochastics will definitely help. For students without this background, Chapter 2 provides a quick summary of the relevant information theory concepts. Introductory-level knowledge of cryptography is useful, simply because this course deals with security. No background in physics is required.

Acknowledgments
Comments by Berry Schoenmakers, Jelle Hellings, Sebastian Banescu, Dion Boesten and Wouter de Koning have helped to improve these lecture notes.

– August 2014
Chapter 2

Information theory

2.1 Probabilities

Probabilities pop up whenever we try to find a mathematical description of something that is not fully predictable (a ‘process’, ‘event’, ‘measurement’, ‘experiment’). Even after careful analysis of such a process, no more knowledge is available about it than a set of possible outcomes and associated probabilities that those outcomes will be realized.

Often the lack of predictability is simply the result of our inability to see (resolve) all relevant details that play a role in the process. For example, we cannot predict the roll of a die because we do not know its exact state of motion when it is released, the exact air density, surface roughness, etc. However, there are important cases where even perfect knowledge of starting conditions does not guarantee predictability. This typically occurs in quantum physics, e.g. when the polarisation of a single light particle is measured.

2.1.1 Random variables

We will use the following notation and concepts when we talk about randomness. A ‘random variable’ (RV) or ‘stochastic variable’ is a quantity whose value depends on the outcome of a nondeterministic process. Random variables are denoted as capital letters. The values that they can take are written as lowercase letters. The notation $\Pr[X = x]$ means; the probability that the random variable $X$ takes the value $x$. Making this distinction between $X$, which is the abstract concept of a process, and $x$, which is a concrete number, is very useful.

The set or space in which a variable lives is often denoted in uppercase calligraphic font, e.g. $\mathcal{X}$. It always holds that the probabilities sum up to 1, i.e. $\sum_{x \in \mathcal{X}} \Pr[X = x] = 1$. The vector of probabilities $(\Pr[X = x])_{x \in \mathcal{X}}$ is called the probability distribution of $X$, or probability mass function (pmf), and notation like $\mathbb{P}$ is sometimes used for it. The notation $X \sim \mathbb{P}$ means: The random variable $X$ has probability distribution $\mathbb{P}$. The notation $\mathbb{P}(x)$ is sometimes used for $\Pr[X = x]$.

**Example 2.1 (Fair coin flip)** Let $X$ describe a fair 0/1 coin flip. We have $X \in \mathcal{X}$, with $\mathcal{X} = \{0, 1\}$, and $X \sim \mathbb{P}$ with $\mathbb{P} = (\frac{1}{2}, \frac{1}{2})$. The probabilities of the outcomes are $\mathbb{P}(0) := \Pr[X = 0] = 1/2$ and $\mathbb{P}(1) := \Pr[X = 1] = 1/2$.

**Continuous random variables (CRVs)**

When $\mathcal{X}$ is a continuum (e.g. $\mathbb{R}$) then the notation $\Pr[X = x]$ does not work, since in general the probability of exactly getting $x$ is zero. Instead one works with a probability density function (pdf),

\[^1\text{from the Greek στοχάστικός = guess}\]
Figure 2.1: Left: The normal distribution. Right: Its cdf.

typically written as \( f_X(x) \), with \( f_X : \mathcal{X} \to [0, \infty) \). For a one-dimensional space the definition is

\[
f_X(x) = \lim_{\varepsilon \downarrow 0} \frac{1}{\varepsilon} \cdot \Pr[x \leq X \leq x + \varepsilon].
\]

(2.1)

In the case of higher-dimensional spaces, the definition is similar. From the pdf, probabilities are obtained by computing integrals,

\[
\Pr[a \leq X \leq b] = \int_a^b f_X(x) \, dx.
\]

(2.2)

The cumulative distribution function (cdf) is defined as \( F_X(x) = \Pr[X \leq x] \). The total probability is always 1, i.e. \( \int_{-\infty}^{\infty} f_X(x) \, dx = 1 \). Obviously there is no big conceptual difference between the discrete case and the continuum case. Pdfs are equivalent to discrete probabilities, and integration is just another kind of summation. However, we will see later that the difference can cause technical difficulties.

**Example 2.2 (Uniform distribution on \([u, v]\))** Consider \( X \in [u, v] \), with all outcomes equally likely. We have \( \mathcal{X} = [u, v] \), and \( f_X(x) = 1/(v - u) \) for \( x \in \mathcal{X} \).

**Example 2.3 (Normal distribution)** The normal (or Gaussian) distribution on \( \mathbb{R} \) is given by

\[
f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.
\]

The cdf is \( F_X(x) = \frac{1}{2} + \frac{1}{2} \text{Erf}(x/\sqrt{2}) \).

One special distribution is the Dirac delta function \( \delta(x) \). It is not an actual function, but the limit of a function. It is zero for \( x \neq 0 \) and \( +\infty \) at \( x = 0 \). It is normalized to 1: for any interval \( J \subset \mathbb{R} \) that contains 0, it holds that \( \int_J dx \, \delta(x) = 1 \). It has a special property just like the Kronecker delta:

\[
\int_{-\infty}^{\infty} dx \, b(x) \delta(x-a) = b(a)
\]

(2.3)

for any function \( b \). In this sense it is the continuum version of the Kronecker delta. The delta function can be written as a limit of an actual function in many ways, for instance

\[
\delta(x) = \lim_{\varepsilon \downarrow 0} \frac{1}{\pi} \frac{\varepsilon}{\varepsilon^2 + x^2}, \quad \delta(x) = \lim_{\varepsilon \downarrow 0} \frac{1}{\varepsilon \sqrt{2\pi}} e^{-x^2/2\varepsilon^2}.
\]

(2.4)

The primitive of the Dirac delta function is called the Heaviside step function \( \Theta(x) \),

\[
\Theta(x) = \int_{-\infty}^{x} dx' \delta(x') \; ; \; \Theta(x) = \begin{cases} 
0 & \text{for } x < 0 \\
1/2 & \text{for } x = 0 \\
1 & \text{for } x > 0 
\end{cases}.
\]

(2.5)

Integration of (2.4) gives

\[
\Theta(x) = \lim_{\varepsilon \downarrow 0} \left[ \frac{1}{2} + \frac{1}{\pi} \arctan \frac{x}{\varepsilon} \right], \quad \Theta(x) = \lim_{\varepsilon \downarrow 0} \left[ \frac{1}{2} + \frac{1}{\varepsilon \sqrt{2\pi}} \text{Erf} \left( \frac{x}{\varepsilon \sqrt{2}} \right) \right].
\]

(2.6)
2.1.2 Expectation value

The expectation value of a quantity is a concept similar to the average value. However, ‘average’ refers to a summation of values that have already been measured, while ‘expectation value’ refers to events in the future. When an experiment has been repeated many times, the average of the outcomes converges to the expectation value.

**Definition 2.4 (Expectation)** Consider a discrete random variable \( X \in \mathcal{X} \). Let \( g(X) \) be a function of \( X \). The expectation value of \( g(X) \) is defined as

\[
E[g(X)] = \sum_{x \in \mathcal{X}} \Pr[X = x]g(x).
\]

**Exercise 2.1** Give the definition of the expectation value for a continuous random variable.

**Example 2.5** Consider a (fair) 4-sided die, \( X \in \{1, 2, 3, 4\} \), with \( \Pr[X = x] = 1/4 \) for all \( x \). We want to know the expectation value of \( X^2 \). The answer is \( \sum_{x=1}^{4} (1/4)x^2 = 15/2 \).

**Example 2.6** In the game of roulette there are 37 equally probable outcomes, \( X \in \{0, 1, \cdots, 36\} \). If you bet on a single number \( x \) and win, your gain is 35 times the money you put in. For this kind of bet, the expectation value of your gain is (expressed as a factor of what you put in)

\[
35 \cdot \Pr[X = x] - 1 \cdot \Pr[X \neq x] = 35 \cdot \frac{1}{37} - 1 \cdot \frac{36}{37} = -\frac{1}{37} \approx -0.027.
\]

**Exercise 2.2** What is the expected duration of a game of Russian roulette?

**Exercise 2.3** Let \( X_1, \cdots, X_n \) be independent random variables that have the same distribution, with \( E[X_i] = \mu \) and \( E[X_i^2] = \mu^2 + \sigma^2 \) for all \( i \). Let \( M := \frac{1}{n} \sum_i X_i \) and \( S^2 := \frac{1}{n} \sum_i (X_i - M)^2 \). Compute the expectation values \( E[M] \) and \( E[S^2] \) in terms of \( \mu \) and \( \sigma \). Do you notice anything strange in your result for \( E[S^2] \)?

2.1.3 Moments

The expectation value \( E[X^k] \) is called the \( k \)’th moment of \( X \) (or of its distribution).

2.1.4 Statistical distance

Let \( X, Y \in \mathcal{X} \) be random variables on the same space but with different distributions, \( X \sim P \) and \( Y \sim Q \). The statistical distance between \( X \) and \( Y \) is defined as

\[
\Delta(X, Y) = \frac{1}{2} \sum_{x \in \mathcal{X}} |P(x) - Q(x)|.
\]

This measure of distance is also called the total variational distance or the \( L_1 \) distance (multiplied by \( \frac{1}{2} \)). It is also sometimes denoted as \( \Delta(P, Q) \) to emphasize that it is actually a distance between distributions, and not between variables.

It can be shown that the statistical distance can also be expressed as

\[
\Delta(X, Y) = \max_{Z \subseteq \mathcal{X}} |\Pr[X \in Z] - \Pr[Y \in Z]|.
\]

**Example 2.7** Consider a fair 6-sided die \( X \sim P = (\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}) \) and a loaded die \( Y \sim Q = (\frac{2}{15}, \frac{2}{15}, \frac{2}{15}, \frac{2}{15}, \frac{2}{15}, \frac{1}{15}) \). Then \( \Delta(X, Y) = \frac{1}{2}[\frac{1}{6} - \frac{2}{15}] + [\frac{1}{6} - \frac{2}{15}] = \frac{1}{6} \).

**Exercise 2.4** Show that \( \Delta(X, Y) \leq 1 \) for any \( X \) and \( Y \). When does it occur that \( \Delta(X, Y) = 1 \)?
2.1.5 Joint distributions; independence; covariance

In many cases, a single stochastic process produces more than one random variable. We use the notation $(X, Y) \sim \mathbb{P}$ to denote that $X$ and $Y$ have joint distribution $\mathbb{P}$. This means that $\mathbb{P}(x, y) = \Pr[X = x, Y = y]$. $X$ and $Y$ do not have to live in the same space. The distribution satisfies $\sum_{x \in X, y \in Y} \mathbb{P}(x, y) = 1$.

Variables are called independent if their joint probabilities factorize, i.e. $\forall x, y \Pr[X = x, Y = y] = \Pr[X = x] \Pr[Y = y]$. This means that $X$ and $Y$ are the result of processes that have no influence on each other. We then have $\mathbb{E}[f(X)g(Y)] = \mathbb{E}[f(X)]\mathbb{E}[g(Y)]$ for any functions $f, g$.

There are many ways to quantify how much interdependence there is between random variables. The simplest measure is the covariance matrix. Let $(X_1, \cdots, X_n)$ be a vector of RVs, then their covariance matrix is defined as

$$K_{ij} = \mathbb{E}[X_i X_j] - \mathbb{E}[X_i] \cdot \mathbb{E}[X_j].$$

(2.9)

This is a symmetric matrix. The diagonal elements are the variance ($\sigma^2$) of each variable separately. It is easy to see that independence of $X_1$ and $X_j$ implies that $K$ is diagonal.

**Example 2.8** Let $X_1$ and $X_2$ be independent, with variances $\sigma_1^2$ and $\sigma_2^2$ respectively. Their covariance matrix is $K = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}$.

**Exercise 2.5** Let $X_1$ and $X_2$ be two variables with zero covariance ($K_{12} = K_{21} = 0$). Does this imply that they are independent? Hint: Consider $X_2 = \alpha X_1^2$.

2.1.6 Marginal distributions

From a joint distribution for $n$ RVs one can derive the distribution of any $k < n$ of the RVs simply by summing over all the others. The result is called a marginal distribution.

**Example 2.10** Let $(X, Y) \sim \mathbb{P}$. Then the marginal distribution for $X$ is

$$\Pr[X = x] = \sum_y \mathbb{P}(x, y).$$

We will (sometimes) denote the marginal distribution in the example above as $\mathbb{P}_1(x) := \Pr[X = x]$, where the ‘1’ stands for the fact that we are selecting the first RV.

2.1.7 Conditional probabilities

In cryptography there are many situations where an attacker has partial knowledge of some secret, e.g. by eavesdropping or side channel attacks. It is important to understand how such knowledge influences the secrecy of the remaining degrees of freedom that are unknown to the attacker.

Consider a joint distribution $(X, Y) \sim \mathbb{P}$. The attacker is always able to see $Y$. What is, from the attacker’s point of view, the distribution of $X$? The answer is

$$\Pr[X = x | Y = y] = \frac{\Pr[X = x, Y = y]}{\Pr[Y = y]} = \frac{\mathbb{P}(x, y)}{\mathbb{P}_2(y)}.\quad (2.10)$$

The ‘$X | Y$’ notation is read as ‘$X$ given $Y$’. It is easily verified that $\sum_x \Pr[X = x | Y = y] = 1$.

Eq. (2.10) can be rewritten as $\Pr[X = x, Y = y] = \Pr[Y = y] \Pr[X = x | Y = y]$, which is a ‘chain rule’ for probabilities. For the events $X = x$ and $Y = y$ to both happen, it is necessary that $Y = y$ happens, and then, on top of that, also $X = x$ given that $Y = y$ has already happened.

**Exercise 2.6** Two six-sides dice are thrown, a red one and a blue one. You are told only that red+blue equals 5. What is the prob. distribution for the blue die, conditioned on this partial knowledge? And for the red die?
2.2 Shannon entropy

2.2.1 Measuring the amount of information

It is very useful to have a good measure for the ‘amount of real information’ in a random variable. It is intuitively clear that, for instance, the roll of a 12-sided die carries more information than a coin flip, and that a biased coin (being more predictable) has less information than a fair coin. Finding a good way of measuring this, however, is not completely trivial. Obviously, ‘predictability’ reduces information. The amount of information grows with the number of possible outcome values of the RV. If all outcomes are equally likely, there is more information than if one of them has a very large probability. From such intuition we can make a list of desirable formal properties:

1. **Additivity**: The information content of a set of independent RVs must be the sum of the individual information contents.

2. **Sub-additivity**: The total information content of two jointly distributed RVs cannot exceed the sum of their separate informations.

3. **Expansibility**: Adding an extra outcome to the set of possible outcomes of the experiment, with probability 0, does not affect the information.

4. **Normalization**: The distribution \((\frac{1}{2}, \frac{1}{2})\) has an information content of 1 bit.

5. The distribution \((p, 1 - p)\) for \(p \neq 0\) has zero information.

It turns out that there is only one measure that satisfies all these requirements. It is called Shannon entropy. Let \(X \in \mathcal{X}\), let \(X \sim \mathbb{P}\) and \(p_x = \mathbb{P}(x)\). The Shannon entropy of \(X\) is defined as

\[
H(X) = \sum_{x \in \mathcal{X}} p_x \log_2 \frac{1}{p_x}. \tag{2.11}
\]

**Exercise 2.7** Show that the Shannon entropy (2.11) has the following properties.

- Properties 1, 3, 4 and 5 above.

- If the distribution is uniform, then \(H(X) = \log_2 |\mathcal{X}|\).

The formula (2.11) can be interpreted as \(\mathbb{E}[\log_2(1/p_x)]\), i.e. the average of all the ‘uniform’ entropies \(\log_2(1/p_x)\).

**Example 2.11** Consider an experiment with two possible outcomes, one with probability \(p\) and the other with probability \(1 - p\). The entropy of the distribution \((p, 1 - p)\) is

\[
h(p) = p \log \frac{1}{p} + (1 - p) \log \frac{1}{1 - p}.
\]

The function \(h\) is called the binary entropy function.
A very useful way of thinking about Shannon entropy is as follows. $H(X)$ is a lower bound on the average number of binary (yes/no) questions that you need to ask about the RV in order to learn the outcome $x$. Every binary question corresponds to one bit of information.

**Example 2.12** Consider a fair 8-sided die. You could first ask if $x$ lies in the range 1–4 or 5–8. The 2nd question could be to ask if $x$ lies in the lowest two values of that range; the 3rd question if $x$ is the lowest value of the remaining two possibilities. This exactly matches $\log_2 8 = 3$.

**Example 2.13** Consider a fair six-sided die. The first question is if $x$ lies in the range 1–3 or 4–6. The second question is if $x$ is the lowest element of this range. With prob. $2/3$, an extra question will be needed. The average is $\frac{1}{2} \cdot 2 + \frac{2}{3} \cdot 3 = \frac{14}{3} \approx 2.67$ questions, which is more than $\log_2 6 \approx 2.58$.

**Exercise 2.8** In example 2.13, what could you do to get closer to the theoretical bound $\log_2 6$?

Similarly, $H(X)$ is a lower bound on the average length of the shortest description of $X$. In other words, an optimal compression algorithm for $X$ will, on average, yield compressions that have length at least $H(X)$.

Yet another way to look at Shannon entropy comes from ‘typical sequences’. Consider a sequence of $n$ independent experiments $X$. The asymptotic equipartition theorem states that most of the probability mass is contained in approximately $2^{nH(X)}$ ‘typical’ sequences, and that these typical sequences have equal probability ($\approx 2^{-nH(X)}$) of occurring.

A slightly easier formulation of the same statement is as follows. Consider a distribution $\mathbb{S} = \{s_1, \ldots, s_q\}$ with $q \in \mathbb{N}$, $s_i = n_i/N$ (where $n_i \in \mathbb{N}$) and $\sum_{i=1}^{q} s_i = 1$. A string $X \in [q]^N$ is randomly generated in such a way that the symbols $X_i$ are independently drawn from $[q]$ according to the distribution $\mathbb{S}$. We define the ‘type’ $T$ as the set of all strings in $[q]^N$ that contain precisely $n_1$ ones, $n_2$ twos, $\ldots$ $n_q$ ‘$q$’s. For any fixed $t \in T$, it then holds that

$$\Pr[X = t] = \prod_{i=1}^{q} s_i^{n_i} = \prod_{i=1}^{q} s_i^{N s_i} = 2^{\sum_{i=1}^{q} N s_i \log s_i} = 2^{-N \mathbb{H}(\mathbb{S})}. \quad (2.12)$$

The probability $\Pr[X = t]$ is the same for all $t \in T$. When $N$ is very large, the probability of $X$ being in $T$ is overwhelming.

**Exercise 2.9** Show that the size of $T$ is $2^{N \mathbb{H}(\mathbb{S}) - O(\log N)}$. Hint: Use Stirling’s approximation $n! \approx \sqrt{2\pi n} (n/e)^n$.

### 2.2.2 Differential Entropy

For a continuous RV $X \sim \rho$ the Shannon entropy is not defined. Instead there is a quantity known as the differential entropy,

$$h_{\text{diff}}(X) = -\int \rho(x) \log \rho(x). \quad (2.13)$$

**Example 2.14** Consider a Gaussian-distributed vector $X \in \mathbb{R}^n$ with covariance matrix $K$ and zero mean. Its probability distribution is

$$\rho(\vec{x}) = \frac{1}{(\sqrt{2\pi})^n \sqrt{\det K}} \exp[-\frac{1}{2} \vec{x}^T K^{-1} \vec{x}]. \quad (2.14)$$

Its differential entropy is

$$h_{\text{diff}}(\vec{X}) = \frac{1}{2} \log[(2\pi e)^n |\det K|]. \quad (2.15)$$
2.2.3 History of the entropy concept

The concept of entropy first arose in thermodynamics, in the equation \( F = U - TS \), where \( F \) is the (Helmholtz) free energy, \( U \) the total energy, \( T \) the temperature and \( S \) the entropy. The free energy is defined as the amount of energy that can be converted to useful work in a closed system at constant temperature and volume.

Boltzmann's statistical mechanics approach led to the understanding that \( S \) is related to the number of degrees of freedom of the particles making up the physical system, or in other words the number of micro-states (complete specification of all particle states) per macro-state (global state, e.g. characterized by only a handful parameters such as volume, pressure, temperature). If \( p_i \) is the probability that the \( i \)’th micro-state occurs, given a fixed macro-state, then

\[
S = k \sum p_i \ln(1/p_i) \approx k \ln \Omega.
\]

Here \( k \) is Boltzmann’s constant and \( \Omega \) is the number of micro-states that yields the same macro-state.

The 2nd law of thermodynamics states that the entropy of a non-equilibrium isolated system will increase. This is a consequence of the fact that large random systems tend to ‘typical' behaviour.

2.3 Kullback-Leibler distance

There is an information-theoretic measure of distance between pmfs, the so-called \textit{relative entropy} or Kullback-Leibler distance. It is asymmetric; it measures how far some distribution \( Q \) is from the true distribution \( P \). The notation is \( D(P||Q) \).

\begin{definition}[Relative entropy]
Let \( P \) and \( Q \) be distributions on \( X \). The relative entropy between \( P \) and \( Q \) is defined as

\[
D(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}.
\]

(2.16)
\end{definition}

It can be shown that \( D(P||Q) \geq 0 \), with equality occurring only when \( P = Q \). Relative entropy has an intuitive meaning: If \( X \sim P \) and you do not know \( P \) but only an approximate distribution \( Q \), then the number of binary questions that you have to ask (on average) to determine \( X \) is \( H(P) + D(P||Q) \).

\begin{exercise}
Prove that \( D(P||Q) \geq 0 \). Hint: make use of \( \mathbb{E} \log(\cdots) \leq \log \mathbb{E}[\cdots] \).
\end{exercise}

2.4 Conditional entropy and mutual information

‘Mutual information’ is one of the most useful concept in information theory, with countless applications in communication theory, cryptography, computer science etc.

First we have to define conditional entropy. For jointly distributed \( X, Y \), the conditional entropy \( H(X|Y) \) measures how much uncertainty there is about \( X \) if you know \( Y \).

\begin{definition}[Entropy of jointly distributed RVs]
Let \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \) be RVs with joint distribution \( P \). Let \( p_{xy} = P(x, y) \). The entropy of \( X \) and \( Y \) together is defined as

\[
H(X,Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{xy} \log \frac{1}{p_{xy}}.
\]

\end{definition}

\begin{definition}[Conditional entropy]
Let \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \) be RVs with joint distribution \( P \). Let \( p_y = \sum_x P(x, y) \) and \( p_{x|y} = P(x, y)/p_y \). The conditional entropy of \( X \) given \( Y \) is defined as

\[
H(X|Y) = \mathbb{E}_y [H(X|Y = y)] = -\sum_{y \in \mathcal{Y}} p_y \sum_{x \in \mathcal{X}} p_{x|y} \log p_{x|y}.
\]


Some caution is necessary here regarding notation. While “$X|Y = y$” is a well behaved probability distribution on $X$, for which the expression $H(X|Y = y)$ makes perfect sense, the notation “$X\{Y$” does not stand for a probability distribution. Rather, $X|Y$ is the set of distributions $\{X|Y = y\}_{y \in \mathcal{Y}}$.

**Exercise 2.11** Prove property 2 (sub-additivity) of the Shannon entropy. Hint: write the expression $H(X,Y) - H(X) - H(Y)$ as a relative entropy.

**Exercise 2.12** Show that $H(X|Y) = H(X,Y) - H(Y)$.

**Exercise 2.13** Show that $H(X|Y) = H(X)$ if $X$ does not depend on $Y$.

**Example 2.18** Consider $X, Y \in \{0, 1\}$, and $(p_{xy}) = \left(\frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{2}{9}, \frac{2}{9}, \frac{2}{9}, \frac{2}{9}, \frac{2}{9}, \frac{2}{9}\right)$. Then $p_{x|y} = \frac{1}{9}$, $p_{y|x} = \frac{2}{9}$, and $p_{x|y}$ is given by $p_{1|0} = p_{0|0} = \frac{1}{2}$, $p_{1|1} = \frac{3}{4}$, $p_{0|1} = \frac{1}{4}$. This gives $H(X|Y = 0) = 1$ and $H(X|Y = 1) = 2 - \frac{1}{2} \log 3$. Averaging, we get $H(X|Y) = \frac{1}{2} \cdot 1 + \frac{3}{4} \cdot (2 - \frac{1}{3} \log 3) = \frac{7}{8} - \frac{1}{8} \log 3 \approx 0.87$.

**Example 2.19** We roll two 6-sided dice. Let $X$ be the sum of the dice. Let $Y \in \{0, 1, 2\}$ be the number of uneven rolls. How much uncertainty is there about $X$? First we look at $Y = 0$. The possible outcomes for $X$ are then $\{4, 6, 8, 10, 12\}$ with probabilities $\left(\frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}\right)$. For $Y = 1$ the possibilities are $\{3, 5, 7, 9, 11\}$, with the same set of probabilities as above. For $Y = 2$ it is $\{2, 4, 6, 8, 10\}$, again with the same probabilities. Hence, no matter what the value of $y$ is, the $p_{x|y}$ is given by $\left(\frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}\right)$. This gives $H(X|Y = y) = \frac{2}{3} \log 2 - 4/9 \approx 2.2$ independent of $y$. Averaging over $y$ is trivial.

For probabilities there is a well known multiplicative chain rule,

$$p(x_1, \ldots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1) \cdots p(x_n|x_{n-1}, \ldots, x_1). \quad (2.17)$$

It leads to an additive chain rule for entropies.

**Lemma 2.20** (Chain rule for entropies)

$$H(X_1, \ldots, X_n) = H(X_1) + H(X_2|X_1) + H(X_3|X_2, X_1) + \cdots + H(X_n|X_{n-1}, \ldots, X_1).$$

**Mutual information** measures the information ‘overlap’ between two variables, i.e. the information that they have in common.

**Definition 2.21** (Mutual information) Let $X, Y$ be RVs. The mutual information of $X$ and $Y$ is

$$I(X; Y) = H(X) - H(X|Y)$$

$$= H(Y) - H(Y|X)$$

$$= H(X,Y) - H(X|Y) - H(Y|X)$$

$$= H(X) + H(Y) - H(X,Y)$$

$$= D(P_{XY}||P_X \times P_Y). \quad (2.18)$$

In a sense the mutual information of $X$ and $Y$ is the opposite of conditional entropy. Given $Y$, together the ‘unknown’ information $H(X|Y)$ and the ‘known’ information $I(X; Y)$ add up to $H(X)$. Note that $I(\cdot; \cdot)$ is symmetric in its arguments, i.e. $I(X; Y) = I(Y; X)$, whereas the conditional entropy does not have this symmetry. Fig. 2.2 graphically shows the relations between all the quantities. The left circle represents $H(X)$. You can think of it as the set of all binary questions that have to be answered before $X$ is known. The right circle is $H(Y)$. Their overlap is $I(X; Y)$. You can think of the overlap as the set of binary questions that yield a bit of information about $X$ and about $Y$. The area $H(X|Y)$ are the questions about $X$ that give no information on $Y$.

$$I(X; Y) = \sum_{xy} p_{xy} \log \frac{p_{xy}}{p_x p_y} \quad (2.19)$$
2.5. RÉNYI ENTROPY

![Figure 2.2: Relation between conditional entropies and mutual information. The total shaded area is $H(X, Y)$.](image)

**Definition 2.22 (Conditional mutual information)** Let $X, Y, Z$ be RVs. The mutual information of $X$ and $Y$, conditioned on $Z$, is defined as

$$I(X; Y|Z) = \mathbb{E}_z I(X|Z = z ; Y|Z = z).$$

(2.20)

**2.5 Rényi entropy**

[Of Section 2.5 only min-entropy will be part of the exam.]

Measures other than Shannon entropy are useful too.

**Definition 2.23 (Rényi entropy)** Let $\alpha \in [0, 1) \cup (1, \infty)$. The Rényi entropy of order $\alpha$ is defined as

$$H_\alpha(X) = \frac{-1}{\alpha - 1} \log \sum_{x \in X} p_x^\alpha.$$

Two special cases are $H_2(X)$, called the collision entropy, and $H_\infty(X)$, called the min-entropy. We will encounter them in Chapter 3.

**Exercise 2.14** Show that $\lim_{\alpha \to \infty} H_\alpha(X) = -\log p_{\text{max}}$, where $p_{\text{max}} = \max_x p_x$.  

**Exercise 2.15** Show that $\lim_{\alpha \to 1} H_\alpha(X) = H(X)$. Hint: Use de l’Hôpital’s rule.

The min-entropy is also called the guessing entropy. It is often denoted as $H_{\min}$. It is (minus) the logarithm of the most likely event; $p_{\text{max}}$ is the probability of guessing $X$ correctly in one go. In crypto it is often prudent to use $H_{\min}$ rather than $H$ as a measure of uncertainty about a key, especially when the attacker can enforce worst case scenarios.

**Example 2.24** Consider $X \in [2^n]$ with $\Pr[X = 1] = \frac{1}{2}$ and $\Pr[X = x] = 1/(2^{n+1} - 2)$ for $x \neq 1$. The distribution has a very high peak; it would be very dangerous to use $X$ as a secret key. The min-entropy is $-\log \frac{1}{2} = 1$. The Shannon entropy, on the other hand, is $\frac{1}{2} + \frac{1}{2} \log (2^{n+1} - 2) \approx n/2$. Hence the Shannon entropy does not ‘see’ the danger posed by the high $p_{\text{max}}$.

**Definition 2.25 (Conditional Rényi entropy)** Let $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ with $(X,Y) \sim \mathcal{P}$. Let $\alpha \in [0, 1) \cup (1, \infty)$ be a constant. The conditional Rényi entropy is defined as $[10]$

$$H_\alpha(X|Y) = \frac{-\alpha}{\alpha - 1} \log \sum_{y \in \mathcal{Y}} p_y \left( \sum_{x \in \mathcal{X}} p_x^\alpha \right)^{1/\alpha}.$$
CHAPTER 2. INFORMATION THEORY

Of particular importance is the limiting case $\alpha \to \infty$, which gives the conditional min-entropy,

$$H_\infty(X|Y) = -\log \sum_{y \in \mathcal{Y}} p_y \max_{x \in \mathcal{X}} p_x|y.$$  \hfill (2.21)

These entropy measures will appear in Chapter 6.

Definition 2.25 may look a bit strange. However, it makes more sense when written in terms of the ‘$\alpha$-norm’. The $\alpha$-norm $|\vec{x}|_\alpha$ of a vector $\vec{x}$ is defined as

$$|\vec{x}|_\alpha := (\sum_i x_i^\alpha)^{1/\alpha}. \hfill (2.22)$$

With this notation, the Rényi entropy of $X$ is written as

$$H_\alpha(X) = \frac{-\alpha}{\alpha - 1} \log |\vec{p}_X|_\alpha$$  \hfill (2.23)

and the conditional Rényi entropy as

$$H(X|Y) = \frac{-\alpha}{\alpha - 1} \log \mathbb{E}_y |\vec{p}_{X|Y=y}|_\alpha.$$  \hfill (2.24)

Also, a more esthetic formula is

$$2^{-\frac{\alpha - 1}{\alpha} H_\alpha(X|Y)} = \mathbb{E}_y 2^{-\frac{\alpha - 1}{\alpha} H_\alpha(X|Y=y)}.$$  \hfill (2.25)

2.6 Error correcting codes

Error correcting codes (ECCs) are closely related to information theory. ECCs play an important role in communication theory and cryptography. In this course we will only consider codes over a binary alphabet and with a linear structure, since these are the simplest.

2.6.1 Linear binary codes

A binary ECC maps a $k$-bit message $x$ to an $n$-bit codeword $c_x$ ($n > k$). This mapping is called encoding. The set of all codewords is denoted as $\mathcal{C}$ and is called the code. The number of codewords is $2^k$. The codewords form a subspace of the space $\{0,1\}^n$. In a linear code, every difference of two codewords is again a codeword: given any $x, y \in \{0,1\}^k$, it holds that $c_x \oplus c_y \in \mathcal{C}$.

The minimum distance $d$ of the code is defined as the minimum Hamming weight between codewords,

$$d = \min_{c,c' \in \mathcal{C}} \text{Hamming weight}(c \oplus c'). \hfill (2.26)$$

Incidentally, since $c \oplus c'$ is a codeword, $d$ is also equal to the minimum Hamming weight among all the codewords.

The codeword is sent over a noisy channel. During transmission some bit errors occur, resulting in a received bit string $z \in \{0,1\}^n$. We will denote the errors as $e$, i.e. $z = c_x \oplus e$. Note that $z$ is not necessarily a codeword! The receiver determines which codeword $c_z$ is closest to $z$. Then he decodes this to $\hat{x}$. If the number of bit errors does not exceed the error correcting capability $t$ of the code,

$$t = \left\lfloor \frac{d - 1}{2} \right\rfloor,$$ \hfill (2.27)

then the decoded message is correct: $\hat{x} = x$. Otherwise a wrong message may be decoded ($\hat{x} \neq x$).

This is intuitively depicted in Fig. 2.3. The smallest possible distance between two codewords is $d$. As long as $z$ is within a radius (Hamming distance) smaller than $d/2$ from $c_x$, there is no ambiguity; $c_x$ is the closest codeword. If the number of bit errors is $d/2$ or larger, then $z$ may be closer to some other codeword, e.g. $c_y$ in the figure.
2.6. ERROR CORRECTING CODES

Figure 2.3: Two codewords with Hamming distance $d$, and the radii within which error correction automatically succeeds.

Something else is also immediately visible from Fig. 2.3. Consider an error pattern with fewer than $d$ errors. The received string is not a codeword. Even if correct decoding fails, it is clear that some bit errors has occurred (even if it not clear to the decoder how many). Hence a code with minimum distance $d$ can detect $d - 1$ bit errors.

The following notation is used for linear binary codes: A code is called an $[n, k, d]$ code if it has $n$-bit codewords, $k$-bit messages and minimum distance $d$. The rate of the code is defined as the fraction of useful payload in a codeword,

$$\text{code rate} = \frac{k}{n}. \quad (2.28)$$

2.6.2 Generator matrix and parity check matrix

In a linear code, codewords can be easily computed using the generator matrix $G$ of the code. The generator matrix of an $[n, k, d]$ code is a $k \times n$ matrix whose $k$ rows are linearly independent codewords. Let $x \in \{0, 1\}^k$ be a message, and interpret it as a row vector. Then the corresponding codeword $c_x$, also interpreted as a row vector, is given by

$$c_x = xG. \quad (2.29)$$

Here addition in the matrix product is modulo 2, i.e. xor. Basically, the rows of $G$ span the subspace $C$.

A generator matrix can always be written in the ‘normal form’,

$$G = (1_k | A). \quad (2.30)$$

(Here $1_k$ stands for the $k \times k$ identity matrix, and the $k \times (n-k)$ matrix $A$ contains the parity check relations.) This form has the advantage that any codeword contains the message in its first $k$ bits, and the rest of the codeword consists of redundancy bits (‘parity check bits’).

Another important matrix is the parity check matrix $H$. This is a $(n-k) \times n$ matrix defined as

$$H = (-A^T | 1_{n-k}). \quad (2.31)$$

It has the special property $GH^T = 0$. Equivalently, for any codeword $c \in C$ we have

$$cH^T = 0. \quad (2.32)$$

**Exercise 2.16** Show that $GH^T = 0$. 
2.6.3 Syndrome decoding

There is a fast way of decoding based on (2.32). It is called ‘syndrome decoding’. When a string \( z = c_x \oplus e \) is received, the so-called syndrome \( s(z) \in \{0, 1\}^{n-k} \) is computed,

\[
s(z) = zH^T = (c_x + e)H^T = eH^T.
\]  

(2.33)

The syndrome depends only on the error pattern.

![Figure 2.5: The equation \( s = zH^T \) with matrix sizes shown.](image)

Use is made of a pre-computed lookup table: for every error pattern \( e_i \) with \#errors \( \leq t \) the syndrome \( s(e_i) \) is tabulated. The size of the table is manageable as long as \( t \) is not too large. The size of the syndromes is \( n - k \) bits. The number of stored syndromes is

\[
\sum_{j=1}^{t} \binom{n}{j}.
\]  

(2.34)

The decoder looks up \( s(z) \) and obtains an error pattern \( \hat{e} \) that satisfies \( s(\hat{e}) = s(z) \). He computes \( c = z + \hat{e} \). The first \( k \) bits of \( c \) are the decoded message. (Remember: in the normal form, the first \( k \) bits of a codeword are just the message.)

Example 2.26 Consider the \([5, 2, 3]\)-code with generator matrix

\[
G = \begin{pmatrix}
1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1
\end{pmatrix}.
\]

The corresponding parity check matrix is

\[
H = \begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1
\end{pmatrix}
\]

The code can correct \( t = 1 \) error. The lookup table is (with syndromes sorted in alphabetical order)

<table>
<thead>
<tr>
<th>Syndrome</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>00001</td>
</tr>
<tr>
<td>010</td>
<td>00010</td>
</tr>
<tr>
<td>100</td>
<td>00100</td>
</tr>
<tr>
<td>101</td>
<td>01000</td>
</tr>
<tr>
<td>110</td>
<td>10000</td>
</tr>
</tbody>
</table>

For these simple error patterns, the syndrome is just equal to a column of \( H \).

2.6.4 A rule of thumb

There exist many different error correcting codes, differing in their code rate, algebraic structure, speed of decoding, memory requirements for decoding etc. There is no generic prescription for e.g. \( n \) as a function of \( k \) and \( t \). However, some bounds can be given.
Theorem 2.27 (Hamming bound) For a binary code of length \( n \) that can correct \( t \) errors, the message length is bounded as
\[
2^k \leq 2^n / \sum_{j=0}^{t} \binom{n}{j}.
\]

Proof: The expression \( \sum_{j=0}^{t} \binom{n}{j} \) is the number of error patterns that the code can correct, (see Eq. 2.34). This is the ‘volume’ of each sphere around a codeword, as depicted in Fig. 2.3. There are \( 2^k \) codewords, so the total volume of all the spheres is \( 2^k \cdot \sum_{j=0}^{t} \binom{n}{j} \). This must not exceed the total volume of the space \( \{0,1\}^n \).

The Hamming bound leads to a somewhat easier-looking approximate bound as follows. For large \( n \) we use the Stirling approximation for \( n! \) and \( \binom{n}{k}! \) and get
\[
k_{\text{max}}(n, t) \approx n - \log(n^t / t!) = n - t\log n + O(t\log t).
\]

This tells us that, as a rule of thumb, a codeword needs approximately \( \log n \) bits of redundancy information per error that you want to correct.

2.6.5 Shannon bound

A communication channel can be characterized by the conditional probability distributions \( \mathbb{P}_Z|C \). You send \( c \) over the channel, and the receiver gets \( z \) with a certain probability \( \mathbb{P}_z|c \). The probability distributions \( \mathbb{P}_z|c \) induce a probability distribution on the error patterns \( z - c \). Consider a code which corrects all the occurring errors with overwhelmingly large probability. It turns out that the rate of such a code \( (k/n) \) is fundamentally limited by the mutual information of the sent and received data. More formally, if \( C \) is a transmitted \( n \)-bit string, and \( Z \) is the received \( n \)-bit string, then the information content \( k \) that can be communicated in an error-free way is bounded by
\[
k \leq I(C; Z).
\]

The simplest case to study is the so-called binary symmetric channel: errors occur independently in each bit of the data, and the probability of a \( 1 \to 0 \) error is the same as for \( 0 \to 1 \). Let us denote the bit flip probability (often called Bit Error Rate, BER) by \( \varepsilon \). Then (2.37) can be expressed per transmitted bit,
\[
\frac{k}{n} \leq I(C_j; Z_j) = H(Z_j) - H(Z_j|C_j)
\]

Since \( Z_j \) is a single bit, its entropy cannot exceed 1. The expression \( H(Z_j|C_j) \) is evaluated using the probability distribution \( p_{Z|C} \). The conditional probabilities are
\[
\Pr[Z_j = c_j | C_j = c_j] = 1 - \varepsilon ; \quad \Pr[Z_j = 1 - c_j | C_j = c_j] = \varepsilon.
\]

This gives
\[
H(Z_j|C_j = c_j) = h(\varepsilon)
\]

independent of \( c_j \). (Here \( h \) is the binary entropy function as defined in Example 2.11.) Averaging over \( c_j \) is trivial. Hence we get
\[
H(Z_j|C_j) = h(\varepsilon).
\]

We conclude that for the binary symmetric channel with BER \( \varepsilon \)
\[
\text{code rate} \leq 1 - h(\varepsilon).
\]
This is in perfect agreement with the rule of thumb (2.36). Using \( \log t! \approx t \log t \), Eq. (2.36) can be rewritten as
\[
\frac{k_{\text{max}}}{n} \approx 1 - \frac{t}{n} \log \frac{n}{t}.
\]
(2.43)
Now taking \( \varepsilon = t/n \) and the small-\( \varepsilon \) approximation of the binary entropy function,
\[
h(\varepsilon) = -\varepsilon \log \varepsilon + \mathcal{O}(\varepsilon),
\]
(2.44)
we see that the approximated Hamming bound (2.43) exactly has the form of the Shannon bound (2.42).

### 2.6.6 Additive Gaussian noise

Consider a \( n \)-component Gaussian RV \( \bar{X} \), sent over a channel with additive Gaussian noise \( \bar{N} \) which is independent of \( X \). The signal at the other side of the channel is \( \bar{Y} = \bar{X} + \bar{N} \), which itself is again Gaussian distributed. We define the correlation matrix \( K_X \) as
\[
(K_X)_{ij} = \mathbb{E}[X_i X_j] - \mathbb{E}[X_i]\mathbb{E}[X_j]
\]
(2.45)
and we define \( K_X \) and \( K_Y \) analogously. We have \( K_Y = K_X + K_N \). The joint distribution of \( X \) and \( Y \) is
\[
\rho(\bar{x}, \bar{y}) = \frac{1}{(\sqrt{2\pi})^{2n} \sqrt{|\det K|}} \exp \left( -\frac{1}{2} \langle \bar{x}, \bar{y} \rangle^T K^{-1} \langle \bar{x}, \bar{y} \rangle \right),
\]
(2.46)
\[
K := \begin{pmatrix} K_X & K_X \\ K_X & K_X + K_N \end{pmatrix} \quad \text{det } K = \text{det } K_X \text{ det } K_N.
\]
(2.47)
Even though the entropy of a continuous RV is undefined (infinite), the mutual information between two continuous RVs is well defined (the infinities cancel). The mutual information is computed using differential entropies,
\[
I(\bar{X}; \bar{Y}) = h_{\text{diff}}(\bar{X}) + h_{\text{diff}}(\bar{Y}) - h_{\text{diff}}(\bar{X}, \bar{Y}).
\]
(2.48)
Using (2.15) we get
\[
I(\bar{X}; \bar{X} + \bar{N}) = \frac{1}{2} \log [(2\pi e)^n | \det K_X |] + \frac{1}{2} \log [(2\pi e)^n | \det (K_X + K_N) |] - \frac{1}{2} \log [(2\pi e)^{2n} | \det (K_X K_N) |]
\]
(2.49)
This is the channel capacity for additive Gaussian noise. In the special case where \( X \) and \( Y \) are scalars, \( K_X \) reduces to the variance \( \sigma_X^2 \) (signal power) and \( K_N \) to the variance \( \sigma_N^2 \) (noise power), yielding
\[
I(X; X + N) = \frac{1}{2} \log [1 + \frac{\sigma_X^2}{\sigma_N^2}].
\]
(2.50)
You may have seen this equation before (or an approximation of it, \( \log \frac{\sigma}{\sigma_N} \)). It says that the signal to noise ratio determines at which rate information can be sent over a channel.
Chapter 3

Random number generation

3.1 Truly random numbers

Truly random numbers (as opposed to pseudo-random) play a central role in many cryptographic protocols. Generating truly random bits is far from trivial. First of all, a physical source of randomness is needed, say some random variable $X$. Most sources do not have a uniformly distributed $X$. Thus, some method is needed to create uniformly distributed bits from it. Preferably it has to be done in such a way that no randomness is ‘wasted’, for in some cases the source is not very good, or a high rate is required. The entropy of the output should then be close to the entropy of $X$. This is already difficult enough when the distribution of $X$ is known; but sometimes it is known only approximately. Implementing a True Random Number Generator (TRNG) in cheap hardware is a difficult engineering problem. Making it resistant against attacks is even more difficult.

3.2 Examples of randomness sources

Ultimately all true randomness originates from quantum physics. The trick is to find some macroscopic process that amplifies the quantum ‘noise’ such that computer hardware can easily measure it. The miniaturization of hardware offers many opportunities here. We list some physical sources that have been studied.

Ring oscillators A ring oscillator consists of an odd number ($n$) of inverters connected in a circular configuration. The output of each inverter oscillates between logical 1 and 0. Ideally, the period of the oscillation would be $T = n\tau$, with $\tau$ the delay time of a single inverter. In reality, there is so-called jitter, random variations of the period. The voltage at some point in the ring can be used as a read-out of the oscillator. Several rings can be combined by XOR-ing their outputs. The combined analog signal lies between some fixed voltages $L$ and $H$ ($L < H$). This is mapped to 0 is it closer to $L$, and to 1 if it is closer to $H$. It has been found empirically that the moment in time when the voltage crosses $(L + H)/2$ has a Gaussian distribution, centered around the ‘ideal’ transition time.

Noisy resistors Even when no external voltage is applied to a resistor, thermal fluctuations cause a small time-dependent random voltage over the resistor. The noise amplitude has an approximately Gaussian distribution. The variance is given by $\langle V^2 \rangle = 4kTR\Delta f$ (Johnson, Nyquist 1928), where $k$ is Boltzmann’s constant ($1.38 \cdot 10^{-23}$J/K), $T$ the temperature, $R$ the resistance, and $\Delta f$ the frequency bandwidth over which the noise is measured. The noise is typically of the order of microvolts.

Radioactive decay In a sense nuclear decay is the ultimate tamper-evident source of randomness. It is impossible to change the decay probability of an atomic nucleus by ordinary
means. A well-built detector can be fooled only by extra irradiation with just the right kind of particles; but this should be detectable during further processing, since the expected decay rate is known.

Given that a nucleus exists at time \( t = 0 \), the probability that it still exists at time \( t \) is given by \( e^{-\lambda t} \), where \( \lambda \) is the decay rate. (The half-life is \( (\lambda \ln 2)^{-1} \).) If there are \( N_0 \) nuclei at time \( t = 0 \), then the number at time \( t \) is expected to be \( N(t) = N_0 e^{-\lambda t} \). The number of decays per unit time is expected to be \( -dN/dt = \lambda N(t) \). Some isotopes have decay products that are easily detectable, e.g. with a Geiger counter. Consider a slowly decaying source, i.e. \( N \) can be treated as a constant. Then the number of 'clicks' of the Geiger counter in a fixed length of time \( \Delta t \) follows a Poisson distribution:

\[
\Pr[\text{#clicks} = k] = e^{-N\lambda \Delta t} \frac{(N\lambda \Delta t)^k}{k!}.
\] (3.1)

Many other systems have been proposed/studied, such as noisy avalanche diodes, metastable flipflops, radio antenna noise, lava lamps, and low-pressure acoustic noise.

Intel’s random number generator uses the amplified thermal noise of undriven resistors. The signal of neighbouring resistors is subtracted to eliminate global effects. The physical process underlying the TRNG may be sensitive to outside influences such as temperature, electromagnetic fields, light, pressure etc. When a TRNG has to operate in an environment controlled by an adversary, special attention must be paid to design the TRNG such that outside effects are filtered out by the measurement circuit, or at least detected in the postprocessing steps.

### 3.3 Making it uniform

#### 3.3.1 Known continuous source

A precisely known continuous source is the easiest case. Whatever the distribution looks like, there is a generic way of turning it into a uniform variable on \([0,1]\). This is possible because there is a whole continuum of coordinate re-parametrisations.

Let \( u \) be a monotonic function, and \( Y = u(X) \). Let \( X \sim f \). We can derive the distribution \( g(y) \) of \( Y \) as follows. First we observe that

\[
\Pr[X \in (x, x + dx)] = \Pr[Y \in (u(x), u(x + dx))]
\] (3.2)

which can be written as

\[
f(x)dx = g(y)dy
\] (3.3)

with \( dy = u'(x)dx \). (See Fig.3.1.) It immediately follows that

\[
g(y) = \frac{f(x)}{u'(x)}.
\] (3.4)

**Exercise 3.1** Do Eqs. (3.2)–(3.4) still hold if \( u \) is not monotonic?

**Theorem 3.1** Let \( X \in \mathbb{R}, X \sim f \), where \( f \) is known exactly. Let \( F \) be the cumulative distribution function. Let \( Y = F(X) \). Then \( Y \) is uniform on \([0,1]\).

**Proof:** We know that \( F(X) \in [0,1] \). Next we want to know the distribution function \( g \) of \( Y \). Since \( Y \) is a monotonic function of \( X \) we can apply (3.4) with \( u(x) \) replaced by \( F(x) \).

\[
g(y) = \frac{f(x)}{F'(x)} = 1.
\] (3.5)
3.3. MAKING IT UNIFORM

Figure 3.1: The probability $f(x)dx$ is identical to $g(y)dy$, with $dy = u'(x)dx$.

Figure 3.2: (a) $N$ intervals of equal area (i.e., probability) under the Gaussian curve $f(x)$, for $N = 8$. The boundary $x_i$ between regions $i - 1$ and $i$ lies at $F(x_i) = i/N$. (b) The corresponding picture, but now for the variable $Y = F(X) \in [0,1]$. The pdf of $Y$ is uniform, and the equiprobable intervals all have equal width $1/N$.

Example 3.2 The normal distribution. Consider $X \sim \mathcal{N}_{\mu,\sigma}$, where $\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[\frac{(x-\mu)^2}{2\sigma^2}\right]$. The cdf is $F(x) = \frac{1}{2} + \frac{1}{2} \text{Erf} \frac{x-\mu}{\sigma \sqrt{2}}$. (Erf is the ‘error function’, the primitive of $e^{-x^2}$.)

Once you have a uniform pdf, it is easy to generate uniform bits. Divide the interval $[0,1]$ into $2^n$ bins of equal width, labeled $q = 0, \ldots, 2^n - 1$. (Fig. 3.2b shows the case $n = 3$.) Do a measurement of $Y$. Look in which bin $y$ sits and output the (binary) label $q(y)$ of that bin. This yields $n$ independent unbiased bits.

3.3.2 Known discrete source; difference with the continuum case

Consider a discrete pmf $\{p_i\}_{i=1}^n$ for some RV $X \in \mathbb{R}$ which can take values $\{x_i\}_{i=1}^n$. You could try to represent the pmf as a fake pdf $f(x)$,

$$f(x) = \sum_{i=1}^{n} p_i \delta(x-x_i), \quad (3.6)$$

1. From $f(x)$ the actual pmf and cdf are obtained as $\Pr[X = x_i] = \lim_{\alpha \to 0} \int_{x_i - \alpha}^{x_i + \alpha} f(x)dx$ and $\Pr[X \leq x_i] = \lim_{\alpha \to 0} \int_{-\infty}^{x_i + \alpha} f(x)dx$. 


and then apply Theorem 3.1 to \( f(x) \), obtaining \( Y = F(X) \),

\[
Y = \sum_{i=1}^{n} p_i \Theta(X - x_i).
\]  

(3.7)

Does this yield a uniform \( Y \)? No! The delta functions in (3.6) are not the smeared-out functions (2.4) before the limit \( \varepsilon \downarrow 0 \), but after the limit. The source \( X \) cannot really take values other than \( x_i \). We end up with a discrete RV \( Y \) with the following pmf,

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( \Pr[Y = y] )</th>
<th>( \Pr[Y \leq y] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( \frac{1}{2} p_1 )</td>
<td>( p_1 )</td>
<td>( p_1 )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>( p_1 + \frac{1}{2} p_2 )</td>
<td>( p_2 )</td>
<td>( p_1 + p_2 )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( x_i )</td>
<td>( p_1 + \cdots + p_i - \frac{1}{2} p_i )</td>
<td>( p_i )</td>
<td>( p_1 + \cdots + p_i )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( x_n )</td>
<td>( 1 - \frac{1}{2} p_n )</td>
<td>( p_n )</td>
<td>( 1 )</td>
</tr>
</tbody>
</table>

Written as a fake continuum pdf \( g(y) \), the pmf of \( y \) is

\[
g(y) = \sum_{i=1}^{n} p_i \delta \left( y - \left[ \frac{i-1}{2} + \frac{p_j}{2} \right] \right),
\]  

(3.8)

with corresponding cumulative distribution \( G(y) \),

\[
G(y) = \sum_{i=1}^{n} p_i \Theta \left( y - \left[ \frac{i-1}{2} + \frac{p_j}{2} \right] \right).
\]  

(3.9)

Does this have any use? Yes! First, notice that \( G(y) \) (Fig. 3.3c) approximates the cdf of the uniform distribution (straight line), snaking around the straight line. The straight line is crossed at the middle of every vertical transition, i.e. at \( y = \sum_{j=1}^{i-1} p_j + \frac{1}{2} p_i \). So in a certain sense the variable \( Y \) is ‘close’ to uniform. This vague statement is made more concrete as follows.

The difference between the original variable \( X \) and our constructed \( Y \) lies in the horizontal gaps between the delta peaks. Compare (3.6) to (3.8). While the weights of the delta peaks are the same \( (p_i) \) in both cases, their location is not: in \( f(x) \) they are completely arbitrary \( (x_i) \), but in \( g(y) \) they are carefully arranged so that every peak has a ‘buffer’ \( p_i/2 \) to its left and to its right. The distance between peak \( i \) and peak \( i + 1 \) is \( \frac{1}{2} p_i + \frac{1}{2} p_{i+1} \). This arrangement does a good job of spreading probability mass over the interval \([0, 1]\), because the horizontal space around a peak is exactly equal to the area (probability) contained in the peak. Thus, when we take a large interval of the \( y \)-axis, the combined area under all the peaks in this interval will be very close to the width of the interval; this is precisely the property of a uniform distribution.

Let us define \( m \) bins of equal width \( (1/m) \), such that \( m \ll n \) and \( \max_i p_i < 1/m \). Then the probability of \( Y \) ending up in such a bin will be almost the same (1/m) for each bin. An example is shown in Fig. 3.3d, with 6 bins. The price we pay is that information has been thrown away. (All events that end up in the same bin have become indistinguishable.) That is the main trade-off for discrete sources: uniformity versus entropy.

### 3.3.3 Known discrete source; some simple examples

**Example 3.3** Consider the pmf \( \{p_1 = \frac{1}{2}, p_2 = \frac{1}{4}, p_3 = \frac{1}{4}\} \). A perfectly uniform pmf can be constructed by binning the 2nd and 3rd event together: \( \{\frac{1}{2}, \frac{1}{2}\} \). The entropy of the uniform pmf is 1 bit, whereas the original source had 1.5 bits of entropy. Note that the min-entropy of the source is 1 bit.
3.3. MAKING IT UNIFORM

Figure 3.3: (a) A non-uniform pmf \( \{p_1, \ldots, p_{20}\} \).
(b) The fake pdf \( g(y) \) for \( Y = F(X) \) according to (3.6).
(c) The fake cdf \( G(y) \). The dotted line is what a uniform distribution would yield.
(d) Binning of \( Y \) into 6 bins of equal width.

In this example the entropy of \( X \) is 4.02 bits; the min-entropy is 3.3 bits; \( X \) has statistical distance 0.28 from uniform; The entropy after binning is 2.577 bits (while \( \log_2 6 \approx 2.585 \)), with statistical distance 0.05 from uniform.

**Example 3.4** Consider the pmf with \( p_1 = p_2 = \frac{1}{4} \) and \( p_3 = p_4 = p_5 = p_6 = \frac{1}{8} \). The Shannon entropy is 2.5 bits and the min-entropy is 2 bits. By grouping \( p_3, p_4 \) together and also \( p_5, p_6 \), the uniform pmf \( \{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\} \) is obtained; its entropy is 2 bits.

**Example 3.5** Consider the pmf \( \{\frac{1}{3}, \frac{1}{6}, \frac{1}{3}, \frac{3}{11}\} \). It has Shannon entropy \( \frac{7}{11} + \frac{5}{11} \log_2 3 + \frac{1}{11} \log_2 5 \approx 1.78 \) bits, and min-entropy \( \log_2 3 \approx 1.58 \) bits. By grouping \( \frac{1}{3} + \frac{1}{6} = \frac{1}{2} \) and \( \frac{1}{3} + \frac{3}{11} = \frac{3}{4} \) we get a perfectly uniform pmf \( \{\frac{1}{2}, \frac{1}{2}\} \) which has 1 bit of entropy. Notice that this is less than the min-entropy.

The min-entropy plays an interesting role here. If we group probabilities as in the examples above, the number of perfectly uniform bits that we can extract is bounded by the min-entropy. This follows from the fact that the bins cannot be smaller than \( p_{\text{max}} \).

However, this does not necessarily say anything about algorithms in which we do not demand perfect uniformity.

**Exercise 3.2** Why is it bad for the uniformity if the bins are made smaller than \( p_{\text{max}} \)?
3.3.4 More generic algorithms

The von Neumann algorithm

Consider a sequence of independent biased coin flips. The von Neumann algorithm removes the bias, but is wasting some of the source’s entropy. The algorithm takes a pair of bits \((b_1, b_2)\) as input, and outputs the following:

\[
\begin{align*}
  b_1 = b_2 & : \text{ no output} \\
  b_1 \neq b_2 & : \text{ output } b_1.
\end{align*}
\] (3.10)

Then the algorithm takes the next two bits of the sequence, etc.

Exercise 3.3  (a) Prove that the output of the von Neumann algorithm is uniform, given that the source bits all have the same bias. 
(b) How much entropy is wasted?

An improvement of the von Neumann algorithm

Several improved de-biasing schemes have been developed. One of them takes four bits as input and outputs either zero, one or two bits. See the table below.

<table>
<thead>
<tr>
<th>input</th>
<th>Neumann</th>
<th>improved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>0010</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>0011</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>0101</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0110</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>0111</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>1001</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>1010</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>1011</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>1100</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>1101</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>1110</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1111</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Exercise 3.4  (a) Prove that the output of the improved von Neumann algorithm is uniform, given that the source bits all have the same bias.  
(b) How much entropy is wasted?

The piling-up lemma

Let \(X_1, \ldots, X_n \in \{0,1\}\) be independent bits with biases

\[
\Pr[X_i = 1] - \Pr[X_i = 0] = \alpha_i. \tag{3.11}
\]

We construct a new bit by xor-ing all the \(X_i\) together: \(Y = X_1 \oplus X_2 \oplus \cdots \oplus X_n\). The bias of \(Y\) is

\[
\Pr[Y = 1] - \Pr[Y = 0] = (-1)^{n-1} \prod_{i=1}^{n} \alpha_i. \tag{3.12}
\]

Hence, by xor-ing multiple independent bits together the bias is reduced. Even if there is only a single \(X_i\) with zero bias, the total bias is zero.

Exercise 3.5  Consider the case \(n = 2\). 
(a) Prove the piling-up lemma.  
(b) How much entropy is wasted?
3.4. PRIVACY AMPLIFICATION USING PUBLIC RANDOMNESS

Resilient functions

This example is motivated by ring oscillators. When the output of an array of ring oscillators is sampled, the signal is unpredictable only during a certain fraction of the time; this is called the “fill rate”, and we will denote it as \( \nu \). Let us assume for simplicity that the output bits of the oscillator array are 100% random during a fraction \( \nu \) of the time, and deterministic for the rest. We also assume that we do not know which of the bits are random. If we want to generate \( m \) random bits, we will have to take at least \( \lceil m/\nu \rceil \) bits of output, and then somehow combine them in a special way so that all the determinism disappears.

Definition 3.6 (Resilient function) A function \( \Psi : \{0,1\}^n \rightarrow \{0,1\}^m \) is called \( (n,m,t) \)-resilient if, for any \( t \) coordinates \( i_1, \ldots, i_t \in [n] \), any \( a_1, \ldots, a_t \in \{0,1\} \) and any \( y \in \{0,1\}^m \) it holds that

\[
\Pr[\Psi(X) = y | x_{i_1} = a_1, \ldots, x_{i_t} = a_t] = 2^{-m}
\]

where \( X \in \{0,1\}^n \) is an RV such that all \( x_j, j \notin \{i_1, \ldots, i_t\} \) are independent and have distribution \((\frac{1}{2}, \frac{1}{2})\).

In words: knowledge of any \( t \) values of the input does not give any information that would help in guessing the output.

When an \( (n,m,t) \)-resilient function \( \Psi \) is applied to a bit string \( x \in \{0,1\}^n \) from the oscillator array, of which \( t \) are deterministic, then \( \Psi(x) \) is a perfectly uniform \( m \)-bit string.

A particularly easy implementation of resilient functions is based on linear error-correcting codes. (See Section 2.6.)

Theorem 3.7 Let \( G \) be the generator matrix of a binary linear \([n,k,d]\) code. Then the function \( \Psi : \{0,1\}^n \rightarrow \{0,1\}^k \) defined as

\[
\Psi(x) = xG^T
\]

is an \( (n,k,d-1) \)-resilient function.

NB: \( \Psi(x) \) is different from the codeword \( c_x = xG \). Also note that a code can correct \( \lfloor (d-1)/2 \rfloor \) errors, but according to Theorem 3.7 gives resilience against \( d-1 \) known input bits.

Proof sketch for Theorem 3.7: The ‘dual’ of a \([n,k,d]\)-code \( C \) is defined as a code \( C^\perp \) with message space \( \{0,1\}^{n-k} \) and codewords of length \( n \) which are orthogonal to \( C \). The parity check matrix of the dual code is given by \( G \). Hence \( \Psi(x) = xG^T \) can be interpreted as a syndrome in the dual code. A syndrome does not depend on the message; the size of the dual message is \( n - k \) bits. Hence \( xG^T \) is insensitive to \( n - k \) bits of \( x \). Finally we note that \( n - k \geq d - 1 \).

3.4 Privacy amplification using public randomness

3.4.1 Universal Hash Functions and the Leftover Hash Lemma

Surprisingly, there exists a generic way to map some \( X \) with barely known pmf to a short bitstring whose pmf is close to uniform. There is a price to pay of course: the closer you want to be to uniform, the shorter the string. Compared to the previous section the entropy loss is very big.

The extraction of a short uniform secret from a non-uniform secret is known as privacy amplification. You start with a variable about which the adversary has some knowledge (e.g. the largest \( p_i \)), and then you hash it down to something about which he knows nothing that can give him any advantage. Hence the ‘privacy’ is ‘amplified’.

One of the essential ingredients is a uniformly random ‘seed’ to aid the extraction. This may look like cheating (our goal is to obtain uniform randomness) but it is not: the seed is publicly known, whereas the extracted string is secret.
Definition 3.8 (Strong extractor) Let \( \text{Ext}: \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^\ell \) be an efficiently computable function that takes as arguments an \( n \)-bit string \( X \) and a random seed \( R \), and outputs an \( \ell \)-bit string \( (\ell < n) \). Let \( Z = \text{Ext}(X, R) \). The function \( \text{Ext} \) is called a strong extractor for source min-entropy \( m \), output length \( \ell \) and nonuniformity \( \varepsilon \) if for all distributions of \( X \) with \( H_m(X) \geq m \) it holds that

\[
\Delta(ZR; U_{\ell}R) \leq \varepsilon. \tag{3.15}
\]

Here \( U_{\ell} \) is an RV uniform on \( \{0,1\}^\ell \). The notation \( \Delta(ZR; U_{\ell}R) \) stands for the statistical distance between on the one hand the joint distribution of \( Z \) and \( R \) and on the other hand the joint distribution of \( U_{\ell} \) and \( R \).

In words, Def. 3.8 says that a strong extractor maps \( X \) to some \( Z \) such that \( Z \) is close to uniform (\( \varepsilon \) stat. distance), even if the seed \( R \) is revealed to the adversary. In the literature the notation \( (m, \ell, \varepsilon) \)-strong extractor is sometimes used.

Exercise 3.6 Show that (3.15) can be written as

\[
\mathbb{E}_r \Delta(Z|R = r; U_{\ell}) \leq \varepsilon. \tag{3.16}
\]

In this form the conditioning on \( R \) is more clearly visible.

Definition 3.9 (Extractable randomness) Let \( X \in \mathcal{X}, Y \in \mathcal{Y} \) be RVs, with \( (X,Y) \sim P \). Let \( R \) be uniformly random on \( \mathcal{R} \). For any \( \varepsilon > 0 \) we say that a finite set \( Z \) is \( \varepsilon \)-allowed if there exists a function \( F: \mathcal{X} \times \mathcal{R} \rightarrow \mathcal{Z} \) such that

\[
\Delta(F(X, R)YR; UYR) \leq \varepsilon,
\]

where \( U \) is a random variable uniformly distributed on \( \mathcal{Z} \), independent of \( X, Y, R \).

The \( \varepsilon \)-extractable randomness of \( X \) conditioned on \( Y \) is defined as

\[
\ell_x^{\text{ext}}(X|Y) = \max \{ \log |Z| : Z \text{ is } \varepsilon\text{-allowed} \}.
\]

You may wonder why the statistical distance is used in Defs. 3.8, 3.9, and not some other measure such as e.g. the KL distance \( D(Z||U_{\ell}) \), or \( H(Z|R) \). As always this has to do with the available proofs for actual constructions that satisfy the definition. There is a class of functions called Universal Hash Functions (UHF) [5] that precisely fits the requirements of a strong extractor. We will give their definition, show that they are useful for privacy amplification, and then give some examples of UHFs.

Definition 3.10 (Universal family of hash functions) Let \( \mathcal{R}, \mathcal{X} \) and \( \mathcal{T} \) be finite sets. Let \( \{\Phi_r\}_{r \in \mathcal{R}} \) be a family of hash functions from \( \mathcal{X} \) to \( \mathcal{T} \). The family \( \{\Phi_r\}_{r \in \mathcal{R}} \) is called universal iff, for \( R \) drawn uniformly from \( \mathcal{R} \), it holds that

\[
\text{Prob}[\Phi_R(x) = \Phi_R(x')] \leq 1/|\mathcal{T}|
\]

for all \( x, x' \in \mathcal{X} \) with \( x' \neq x \).

(The collision probability \(|\mathcal{T}|^{-1}\) is what you would get from a random oracle.) We will see later on that it can have some advantages to relax the definition a little bit. Hash functions that allow a bit higher collision probability are called almost universal [30].

Definition 3.11 (Almost universal family of hash functions) Let \( \eta \geq 0 \) be a constant. Let \( \mathcal{R}, \mathcal{X} \) and \( \mathcal{T} \) be finite sets. Let \( \{\Phi_r\}_{r \in \mathcal{R}} \) be a family of hash functions from \( \mathcal{X} \) to \( \mathcal{T} \). The family \( \{\Phi_r\}_{r \in \mathcal{R}} \) is called \( \eta \)-almost universal iff, for \( R \) drawn uniformly from \( \mathcal{R} \), it holds that

\[
\text{Prob}[\Phi_R(x) = \Phi_R(x')] \leq \eta
\]

for all \( x, x' \in \mathcal{X} \) with \( x' \neq x \).
Note that a $1/|\mathcal{T}|$-almost universal family of hash functions is universal. From the existence of (almost) universal hash functions it can be proven that for some combinations of parameters there always exists a strong extractor. There is a famous statement known as the leftover hash lemma.

**Theorem 3.12 (Leftover hash lemma)** Let $X \in \mathcal{X}$ be a random variable. Let $\delta \geq 0$ be a constant. Let $F : \mathcal{X} \times \mathcal{R} \to \{0,1\}^{\ell}$ be a $2^{-\ell}(1+\delta)$-almost universal family of hash functions, with seed $R \in \mathcal{R}$. Then

$$\Delta(F(X,R); U_{\ell}R) \leq \frac{1}{2}\sqrt{\delta + 2^{\ell - H_2(X)}}, \quad (3.17)$$

where $H_2(X)$ is the Rényi entropy as defined in Section 2.5.

**Corollary 3.13** The $\varepsilon$-extractable randomness of $X$ given $Y$ can be bounded as

$$\ell_{\text{ext}}^\varepsilon(X|Y) \geq H_2(X|Y) + 2 - 2\log \frac{1}{\varepsilon}, \quad (3.18)$$

where $H_2(X|Y)$ is the conditional Rényi entropy (Definition 2.25).

**Proof of Corollary 3.13:** We start from (3.17) with $\delta = 0$, because $\delta = 0$ is optimal for extraction. We consider the variable $X|Y = y$, i.e. $X$ at given $y$. This gives $\Delta(F(X|Y = y,R); U_{\ell}R) \leq \frac{1}{2}\sqrt{2^{\ell - H_2(X|Y = y)}} = \frac{1}{2}\sqrt{2^{\ell/2}} \sqrt{\sum_x p_{x|y}^2}$. We apply the expectation $\mathbb{E}_y$ to both sides of the inequality. The left hand side yields $\mathbb{E}_y \Delta(F(X|Y = y,R); U_{\ell}R) = \Delta(F(X,R)|Y; U_{\ell}R)$, i.e. the statistical distance between $F(X,R)$ and uniformity, at given $R$ and $Y$. The right hand side is $\frac{1}{2}\sqrt{\frac{2^{\ell/2}}{\mathbb{E}_y} \sqrt{\sum_x p_{x|y}^2}} = 2^{-1+\ell/2-\frac{1}{2}H_2(X|Y)}$. So we have obtained

$$\Delta(F(X,R)|Y; U_{\ell}R) \leq 2^{-1+\ell/2-\frac{1}{2}H_2(X|Y)}. \quad (3.19)$$

Now we put the statistical distance $\Delta(\cdots;\cdots)$ to $\varepsilon$. This gives us the largest possible $\ell$ such that the extraction using UHF works. Multiplying both sides of the equation by 2 and then squaring gives $4\varepsilon^2 = 2^{\ell - H_2(X|Y)}$. Taking the logarithm on both sides and then solving for $\ell$ yields $\ell = H_2(X|Y) + 2 - 2\log(1/\varepsilon)$. There is a rather unpleasant property of (3.18). The ‘penalty’ term $-2\log(1/\varepsilon)$ depends on the target uniformity, not on the improvement gained from hashing!

[The rest of Section 3.4.1 is a proof of the Leftover Hash Lemma, and is not part of the exam.]

The proof of Theorem 3.12 is rather long, so we set it up in stages. We start with a number of lemmas that will be invoked by the proof.

**Lemma 3.14 (Jensen’s inequality for concave functions)** Let $\varphi$ be a real concave function. Let $n$ be a positive integer. Let $a_1, \ldots, a_n$ be positive weights and $x_1, \ldots, x_n$ be real numbers. Then

$$\varphi\left(\frac{\sum_{i=1}^n a_i x_i}{\sum_{i=1}^n a_i}\right) \geq \frac{\sum_{i=1}^n a_i \varphi(x_i)}{\sum_{i=1}^n a_i}. \quad \text{(Jensen’s inequality)}$$

**Lemma 3.15** Let $q_1, \ldots, q_m$ be real numbers satisfying $\sum_{s=1}^m q_s = 1$. Then it holds that

$$\sum_{s=1}^m q_s^2 \geq \frac{1}{m}. \quad \text{(Cauchy-Schwarz inequality)}$$

**Proof:** $0 \leq \sum_s (q_s - \frac{1}{m})^2 = \sum_s q_s^2 - \frac{2}{m} \sum_s q_s + \sum_s 1/m^2 = \sum_s q_s^2 - \frac{1}{m}. \quad \Box$

**Lemma 3.16** Let $Z \in \mathcal{Z}$ be a RV. Then the statistical distance between $Z$ and $U$ uniform on $\mathcal{Z}$ can be bounded as

$$\Delta(Z,U) \leq \frac{1}{2} \sqrt{|\mathcal{Z}| \sum_{z \in \mathcal{Z}} p_z^2 - 1}. \quad (3.20)$$
Proof: Define $D = \Delta(Z, U)$ and $q_z := |p_z - 1/|Z||/2D$, satisfying $\sum_z q_z = 1$. Application of Lemma 3.15 yields $\frac{1}{|Z|^2} \leq \frac{1}{4D^2} [\sum_z p_x^2 - \frac{1}{|Z|} \sum_z p_z + \sum_z |Z|^{-2}] = \frac{1}{4D^2} [\sum_z p_x^2 - \frac{1}{|Z|}]$. Rearranging the inequality gives (3.20). □

Proof of Theorem 3.12: First we will prove a simpler version without the conditioning,

$$
\Delta(F(X, R) R; U_t R) \leq \frac{1}{2} \sqrt{\delta + 2^{\ell-H_2(X)}}.
$$

(3.21)

We use Lemma 3.16 replacing $Z \rightarrow F(X, R) R$ and $U \rightarrow U_t R$. The collision probability is

$$
c = \sum_{r,t} (\Pr[R = r, F(X, R) = t])^2 = \sum_{r,t} \left( \frac{1}{|R|} \sum_x [F(x, r) = t] p_x \right)^2
$$

$$
= |R|^{-1} \sum_{x,x'} \sum_{t} \frac{1}{|R|} [F(x, r) = t][F(x', r) = t]
$$

$$
= |R|^{-1} \sum_{x,x'} p_x p_{x'} \Pr[F(x, R) = F(x', R)]
$$

$$
\leq \frac{1}{|R|} \sum_x p_x^2 + \frac{2^{-\ell}(1 + \delta)}{|R|} \sum_{x,x':x \neq x'} p_x p_{x'}
$$

$$
\leq \frac{1}{|R|} \sum_x p_x^2 + \frac{2^{-\ell}(1 + \delta)}{|R|} \frac{1}{|R \times \{0, 1\}|} \left( 2^\ell \sum_x p_x^2 + 1 + \delta \right)
$$

$$
= \left( \frac{1}{|R|} \right) \left( 2^{\ell-H_2(X)} + 1 + \delta \right). \tag{3.22}
$$

Substitution into (3.20) indeed gives (3.21). Now we go to the conditional case. We have

$$
\Delta(F(X, R) Y R; U_t Y R) = \sum_y p_y \Delta(F(X|Y = y, R) R; U_t R). \tag{3.23}
$$

Now we apply (3.21) to the RV $X|Y = y$,

$$
\Delta(F(X, R) Y R; U_t Y R) \leq \sum_y p_y \frac{1}{2} \sqrt{\delta + 2^\ell \sum_x p_x^2 |y|}. \tag{3.24}
$$

Finally, by Jensen’s inequality (Lemma 3.14) we get

$$
\Delta(F(X, R) Y R; U_t Y R) \leq \frac{1}{2} \sqrt{\sum_y p_y \sqrt{\sum_y p_y [\delta + 2^\ell \sum_x p_x^2 |y|]}} = \frac{1}{2} \sqrt{\delta + 2^\ell \sum_y p_x^2 |y|}. \tag{3.25}
$$

As a final remark about the Leftover Hash Lemma, we mention that there is a variant [2] which does not aim for unconditional security; if one is willing to accept a lower level of security, compatible with standard assumptions in cryptography, then the factor 2 in the penalty term $2 \log \frac{1}{\varepsilon}$ in (3.18) can be reduced to a factor 1. □
3.4. PRIVACY AMPLIFICATION USING PUBLIC RANDOMNESS

3.4.2 Examples of (almost) universal hash functions

[Section 3.4.2 is not part of the exam.]

UHFs can be very simple, e.g. based almost entirely on linear operations. That may come as a surprise, but remember that the purpose of a UHF is not to create a cryptographic hash (with pre-image resistance etc.) but only to smoothen out the collision probabilities.

Example 3.17 (UHF) Let \( p \) be a prime, and \( \ell \) a positive integer. Let \( \mathcal{X} = \mathbb{Z}_{p}^{\ell+1} \), \( \mathcal{Z} = \mathbb{Z}_{p} \) and \( \mathcal{R} = \mathbb{Z}_{p}^{\ell} \). We define a function \( \Phi : \mathcal{X} \times \mathcal{R} \rightarrow \mathcal{Z} \), \( (x, r) \mapsto z \) as follows,

\[
\begin{align*}
x & = (x_0, \ldots, x_\ell) \\
r & = (r_1, \ldots, r_\ell) \\
z & = x_0 + x_1 r_1 + \cdots + x_\ell r_\ell \mod p.
\end{align*}
\]

(3.26)

When \( r \) is taken to be the seed, \( \Phi \) is a universal family of hash functions.

Example 3.18 (AUHF) Let \( p \) be a prime, and \( \ell \) a positive integer. Let \( \mathcal{X} = \mathbb{Z}_{p}^{\ell+1} \), \( \mathcal{Z} = \mathbb{Z}_{p} \) and \( \mathcal{R} = \mathbb{Z}_{p}^{\ell} \). We define a function \( \Phi : \mathcal{X} \times \mathcal{R} \rightarrow \mathcal{Z} \), \( (x, r) \mapsto z \) as follows,

\[
\begin{align*}
x & = (x_0, \ldots, x_\ell) \\
z & = x_0 + x_1 r_1 + x_2 r_2^2 + \cdots + x_\ell r_\ell^\ell \mod p.
\end{align*}
\]

(3.27)

When \( r \) is taken to be the seed, \( \Phi \) is a \( \ell/p \)-almost universal family of hash functions.

Example 3.19 (AUHF) Let \( p \) be a prime, and \( \ell \) a positive integer. Let \( \mathcal{X} = \mathbb{Z}_{p}^{\ell} \), \( \mathcal{Z} = \mathbb{Z}_{p} \) and \( \mathcal{R} = \mathbb{Z}_{p} \times \mathbb{Z}_{p} \). We write \( r = (r_0, r_1) \in \mathcal{R} \). We define \( \Phi : \mathcal{X} \times \mathcal{R} \rightarrow \mathcal{Z} \), \( (x, r) \mapsto z \) as follows,

\[
\begin{align*}
z & = r_0 + x_1 r_1 + x_2 r_1^2 + \cdots + x_\ell r_1^\ell \mod p.
\end{align*}
\]

(3.28)

When \( r \) is taken to be the seed, \( \Phi \) is a \( \ell/p \)-almost universal family of hash functions.

Example 3.20 This is an example of a UHF specifically designed to be efficiently implemented in hardware [36]. Let \( w \) be a positive integer and let \( \ell \) be a positive even integer. We work with the Galois field \( GF(2^w)[a] \), i.e. polynomials of the variable \( a \) of order \( \leq w \) with coefficients in \( GF(2) \).

Let \( P \) be an irreducible polynomial over \( GF(2) \) of degree \( w \). Let \( \mathcal{X} = [GF(2^w)]^\ell \), \( \mathcal{R} = [GF(2^w)]^\ell \) and \( \mathcal{Z} = GF(2^w) \). We write \( r = (r_1, \ldots, r_\ell) \), \( x = (x_1, \ldots, x_\ell) \), with \( r_i, x_i \in GF(2^w) \). We define \( \Phi : \mathcal{X} \times \mathcal{R} \rightarrow \mathcal{Z} \), \( (x, r) \mapsto z \) as

\[
\begin{align*}
z = \sum_{i=1}^{\ell/2} (x_{2i-1} + r_{2i-1})(x_{2i} + r_{2i})a^{(\ell/2)w} \mod P.
\end{align*}
\]

(3.29)

Apart from privacy amplification, UHFs play an important role in Message Authentication Codes (MACs). A nice treatment of UHFs can be found in [29].
Exercise 3.7 Consider a True Random Number Generator which consists of a physical source that generates independent biased bits with \( p \approx 0.35 \), followed by the improved von Neumann algorithm. An attacker wants to influence the predictability of the output. He somehow manages to change the bias to \( p - \Delta p \).
(a) What has the attacker achieved in terms of predictability?
(b) Can the attack be detected?

Exercise 3.8 Let \( w \) be some function. Show that \( H(w(X)) \leq H(X) \).

Exercise 3.9 In the definition of a strong extractor we have \( Z = \text{Ext}(X, R) \). Show that \( H(Z|R) \leq H(X) \).

Exercise 3.10 Given is a random variable \( X \in \{0,1\}^n \), such that all bits in the string \( X \) are independent and have probability \( (1 + \alpha)/2 \) of being 1.
(a) Apply Corollary 3.13 to compute the minimum extractable randomness of \( X \) for some arbitrary \( \varepsilon \).
(b) Compare your result to the Shannon entropy of \( X \). Take \( |\alpha| \ll 1 \) and use Taylor expansions to order \( \alpha^2 \). How much entropy is lost when the generic extraction method of UHF's is applied?

Exercise 3.11 The Leftover Hash Lemma gives a bound on the statistical distance from the uniform distribution, but cryptographers are often more interested in the min-entropy of a key.
(a) Let \( Z = F(X, R) \in \{0,1\}^\ell \) be the key. Take \( \delta = 0 \) in Theorem 3.12. Show that
\[
H_{\infty}(Z) \geq -\log \left( \frac{1}{2^{\ell/2}} e^{-H_2(X|Y)/2} + 2^{-\ell} \right).
\]
Hint: For a given statistical distance \( \Delta \), find the worst-case configuration of probabilities, i.e. one that minimizes \( H_{\infty}(Z) \).
(b) Find the length \( \ell \) that optimizes the guaranteed min-entropy of \( Z \). Also give the lower bound on \( H_{\infty}(Z) \) for that value of \( \ell \).
Chapter 4

Key agreement from correlated randomness

4.1 Exploiting noise

It is well known that a cipher can be information-theoretically secure (unconditionally secure) only if
\[ H(K) \geq H(M), \]
where \( K \) is the key and \( M \) is the message. Roughly speaking, the key has to be at least as long as the message. One example of a perfect cipher is the one-time pad (OTP), where the ciphertext is computed by a simple XOR: \( C = K \oplus M \) and \( M = C \oplus K \).

It follows that Alice cannot communicate a message \( M \) to Bob in an unconditionally secure way if they share a short secret key \( K \), with \( H(K) \ll H(M) \). This seems to be a solid, undeniable impossibility theorem. And yet... In 1993 Maurer found a way around it [22]. The proof of the impossibility holds when the data intercepted by Eve is literally equal to the messages between Alice and Bob. In other words, the standard assumption is that the communication as well as the eavesdropping is noiseless. It had already been realized before 1993 that the impossibility theorem may be avoided if Eve's data is noisy. However, early ideas required Eve's channel to be noisier than the channel between Alice and Bob; this is a very dangerous assumption.

We will consider two scenarios:

1. **Noisy broadcast scenario**: Alice has a way of sending bits to Bob over a noisy broadcast channel. For Bob the noise is modeled as a bit error rate \( \varepsilon \), for Eve the noise is \( \delta \). It is quite possible that \( \delta < \varepsilon \). Alice and Bob can also talk to each other over a noiseless (but insecure) channel. The main trick is to define a 'virtual' channel from Bob to Alice, with the special property that Eve's noise on this channel is larger than Alice's.

2. **Satellite scenario**: There is a physical source of randomness which broadcasts random data, e.g. random bits. Alice, Bob and Eve all receive the broadcast with varying degrees of noise, e.g. bit error probabilities \( \varepsilon_A, \varepsilon_B \) and \( \varepsilon_E \) respectively. It is not excluded that Eve has less noise than Alice and Bob! Let's say Alice receives \( X \), Bob \( Y \), and Eve \( Z \). These have joint probability distribution \( P_{XYZ} \). Alice and Bob can also talk to each other over a noiseless (but insecure) channel. The main trick is that Alice and Bob can do error correction (this will be explained later) and privacy amplification by public discussion, based on the mutual information between \( X \) and \( Y \) conditioned on the fact that Eve knows \( Z \), \( I(X;Y \mid Z) \). The error correction, if done properly, will produce bits \( f \) that Alice and Bob agree on without revealing their value to Eve. (Though Eve has good chances of guessing correctly through her knowledge of \( Z \).) Due to the fact that Eve's noise is different from Alice and Bob's, Eve's guess of \( f \) will not always be correct; hence Alice and Bob have some secret bits that they can then extract using privacy amplification.
CHAPTER 4. KEY AGREEMENT FROM CORRELATED RANDOMNESS

Figure 4.1: The satellite scenario. Alice, Bob and Eve receive bits from a satellite with error probabilities $\varepsilon_A$, $\varepsilon_B$, $\varepsilon_E$ respectively. It cannot be excluded that Eve has better reception than Alice and Bob.

A crucial point in both scenarios is that discussion between Alice and Bob over an insecure (public) channel can improve the length of the secret they generate, or even enable such a secret to exist at all when Eve has superior reception.

In this chapter we will study the simple case where the noise is modeled as a binary symmetric channel. Some theorems about the secrecy capacity are proven, and schemes are presented for both scenarios.

4.2 The binary symmetric channel

The binary symmetric channel (BSC) is defined as follows. When a bit is sent over the channel, the probability of a bit flip is $\varepsilon$, independent of previously sent bits. The error probability does not depend on the value of the bit (0/1), hence the name symmetric.

When two such channels, characterized by error rates $\varepsilon_1$, $\varepsilon_2$, are concatenated, the combined error rate is given by

$$\varepsilon_1 \ast \varepsilon_2 := \varepsilon_1 (1 - \varepsilon_2) + \varepsilon_2 (1 - \varepsilon_1) = \varepsilon_1 + \varepsilon_2 - 2\varepsilon_1 \varepsilon_2.$$  (4.2)

(It is easily verified that $\varepsilon_1 \ast 0 = \varepsilon_1$ and $\varepsilon_1 \ast \frac{1}{2} = \frac{1}{2}$, as expected.)

The process of transmitting data over a BSC is modeled as follows. The input is a binary RV $X$. The noise is a binary RV $E$ with $\Pr[E = 1] = \varepsilon$. The received bit at the other side of the channel is $Y = X \oplus E$.

The maximum achievable amount of information that can be reliably sent over the channel, counted per transmitted bit, is called the capacity of the channel. For binary $X, Y$ it is given by $I(X; Y)$.

Exercise 4.1 Consider a uniformly random bit $X$ sent over a BSC with error rate $\varepsilon$. The receiver gets $Y$. Show that $I(X; Y) = 1 - h(\varepsilon)$.

Exercise 4.2 Consider the model of Fig. 4.1. Alice takes a random bit $U$ and broadcasts $U \oplus X$ on a noiseless public channel. Show that Bob can reconstruct $U$ with error probability $\varepsilon := \varepsilon_A \ast \varepsilon_B$ and Eve with error probability $\delta := \varepsilon_A \ast \varepsilon_E$.

4.3 Scenario 1: Noisy broadcast by Alice

4.3.1 Noisy broadcast channel (without return channel)

Alice is broadcasting bits to Bob over a BSC with error rate $\varepsilon \leq \frac{1}{2}$. She sends $X$, and Bob receives $Y$. Eve the eavesdropper receives a noisy version $Z$. The channel from Alice to Eve is a BSC with error rate $\delta \leq \frac{1}{2}$. The channels are jointly described by a distribution $P_{YZ|X}$. There
4.3. SCENARIO 1: NOISY BROADCAST BY ALICE

are no other channels. Bob cannot talk to Alice. Can Alice and Bob generate a secret by using the noise in the channel as a resource? If so, at what rate can they generate such a secret?

**Definition 4.1** Let $C_{RN}$ be the set of error correcting codes with rate $R$ and codeword length $N$. Let the message length be $K = [RN]$. Let a code be denoted as $(e, d)$, with $e$ the encoding function and $d$ the decoding function. The **secrecy capacity** $C_s$ of the broadcast channel $P_{YZ|X}$ is defined as the maximal rate $R$ such that for all $\gamma > 0$ and sufficiently large $N$, there exists a code $(e, d) \in C_{RN}$ satisfying

1. $\Pr[d(Y^N) \neq V] < \gamma$ where $X = e(V)$ and $V$ is uniformly distributed on $\{0,1\}^K$.
2. $\frac{1}{R} H(V|Z^N) > 1 - \gamma$.

In words, the secrecy capacity $C_s$ is defined as the maximum possible rate at which Alice can send codewords to Bob such that (1) Bob correctly decodes with arbitrarily high probability and (2) Eve’s ignorance about the message (computed per message bit), given the bits $Z^N$ that she eavesdropped, can be made arbitrarily close to 100%.

**Theorem 4.2 (Csiszar and Körner 1978)** The secrecy capacity of a binary broadcast channel $P_{YZ|X}$ can be bounded as

$$C_s(P_{YZ|X}) \geq \max_{P_X} [I(X;Y) - I(X;Z)]$$

$$= \max_{P_X} [H(X|Z) - H(X|Y)].$$

A necessary condition for the equality to hold is $I(X;Y) \geq I(X;Z)$ for all $P_X$.

It turns out that the condition $I(X;Y) \geq I(X;Z)$ for all $P_X$ is also a sufficient condition for most distributions of interest. The secrecy capacity is zero unless $I(X;Y) > I(X;Z)$ for some $P_X$. (We do not give a proof.) Intuitively, Theorem 4.2 says that the secrecy rate is determined by the capacity from Alice to Bob minus the capacity from Alice to Eve. If Eve’s reception quality is better than Bob’s, then the secrecy capacity is zero.

**Theorem 4.3** The secrecy capacity of the above described noisy broadcast channel (BSC with error rates $\varepsilon$ and $\delta$) is

$$C_s(\varepsilon, \delta) = \begin{cases} 
  h(\delta) - h(\varepsilon) & \text{if } \delta > \varepsilon \\
  0 & \text{otherwise}
\end{cases}$$

**Proof:** We use Theorem 4.2. Define $\alpha = \Pr[X = 1]$. We have

$$I(X;Y) = H(Y) - H(Y|X) = h(\alpha \ast \varepsilon) - h(\varepsilon)$$

$$I(X;Z) = H(Z) - H(Z|X) = h(\alpha \ast \delta) - h(\delta).$$

(4.3)

This gives

$$\max_{\alpha} [I(X;Y) - I(X;Z)] = h(\delta) - h(\varepsilon) + \max_{\alpha} [h(\alpha \ast \varepsilon) - h(\alpha \ast \delta)].$$

(4.4)

For $\delta > \varepsilon$ we find that the maximum in (4.4) is achieved at $\alpha = \frac{1}{2}$, yielding $h(\frac{1}{2} \ast \varepsilon) - h(\frac{1}{2} \ast \delta) = 0$. (This follows from the fact that for any $\beta \leq \frac{1}{2}$, the function $h(\alpha \ast \beta)$ is an increasing function of $\alpha$ on $[0, \frac{1}{2}]$.)
4.3.2 Noisy broadcast channel plus public return channel

Now we allow Alice and Bob to talk over a noiseless public channel. (Perhaps) surprisingly, this greatly improves the secrecy capacity.

**Theorem 4.4 (Maurer 1993)** The secrecy capacity with public discussion of a noisy broadcast channel $P_{YZ|X}$ is upper bounded by

$$\hat{C}_s(P_{YZ|X}) \leq \min \left\{ \max_{P_X} I(X;Y), \max_{P_X} I(X;Y|Z) \right\}.$$  

(We do not give a proof.) The $I(X;Y)$ appears in the theorem because there exist some distributions $P_{XYZ}$ such that $I(X;Y|Z) > I(X;Y)$. I.e., while conditioning always reduces entropy, mutual information may actually be increased in some cases. The theorem has a form that you would have guessed intuitively: $I(X;Y|Z)$ is what is left of the mutual information between $X$ and $Y$ once Eve’s knowledge has been subtracted.

**Theorem 4.5** The secrecy capacity with public discussion of the described binary symmetric broadcast channel with error rates $\epsilon, \delta$ is given by

$$\hat{C}_s(\epsilon, \delta) = h(\epsilon \ast \delta) - h(\epsilon).$$

**Corollary 4.6** $\hat{C}_s(\epsilon, \delta) > 0$ unless $\epsilon = \frac{1}{2}$ or $\delta = 0$.

**Proof of Theorem 4.5**: We first prove a lower bound on $\hat{C}_s(\epsilon, \delta)$ by describing a scheme that achieves $h(\epsilon \ast \delta) - h(\epsilon)$. Then we use Theorem 4.4 to show that it is equal to the upper bound. 

The scheme. We create a conceptual channel from Bob to Alice as follows. Alice sends a bit $X$ over the broadcast channel, with $\Pr[X = 1] = \frac{1}{2}$. Bob receives $Y = X \oplus E$, Eve receives $Z = X \oplus D$, where $E, D \in \{0, 1\}$ are the errors. Bob takes a random bit $V$ and sends $W = Y \oplus V$ to Alice over the public channel. Alice computes $W \oplus X = V \oplus E$, i.e. she receives Bob’s bit with error probability $\epsilon$. Eve, on the other hand, has $W$ and $Z$, from which she computes $W \oplus Z = Y \oplus V \oplus X \oplus D = V \oplus E \oplus D$. Hence Eve receives $V$ through a concatenation of the two noisy channels: for her the noise on $V$ is $\epsilon \ast \delta$. (In exercise 4.3 you are asked to prove that Eve loses no information about $V$ if she keeps $W \oplus Z$ and discards $W, Z$.) The lower bound on $\hat{C}(\epsilon, \delta)$ follows from the fact that Bob can use the conceptual broadcast channel to transmit a secret to Alice, yielding secrecy rate $h(\epsilon \ast \delta) - h(\epsilon)$ according to Theorem 4.3.
4.4. SCENARIO 2: THE SATELLITE SCENARIO

The upper bound. We have

\begin{align*}
I(X;Y|Z) &= H(Y|Z) - H(Y|XZ) \\
&= H(Y|Z) - H(Y|X) \\
&= I(Y;X) - I(Y;Z).
\end{align*}

(4.5)

In the second step we used that the channels are independent. Now we apply Theorem 4.4, noting that \(I(Y;Z)\) is the capacity of the conceptual channel. We then do the optimization of \(P_X\) as in the proof of Theorem 4.3, finding the maximum at \(\Pr[X = 1] = 1/2\), and in that maximum the secrecy capacity \(h(\varepsilon * \delta) - h(\varepsilon)\).

\[\square\]

Exercise 4.3 For the proof of Theorem 4.5, show that \(H(V|WZ) = H(V|W \oplus Z)\).

Exercise 4.4 Prove corollary 4.6.

We have seen that, surprisingly, the secrecy rate is still nonzero even if Eve’s noise is close to zero! Hence it is possible for Alice and Bob to generate a secret key, exploiting Eve’s noise. They can then use the secret key as a one-time pad. In this way the pessimistic requirement (4.1) is avoided. So, how do Alice and Bob generate a secret key? The answer is contained in Def. 4.1. Bob sends codewords to Alice from a code with parameters chosen precisely so that a bit error rate of \(\varepsilon\) can be corrected but \(\delta * \varepsilon\) cannot.

\[\text{Remark: A short initial MAC key may be needed to authenticate the noiseless public messages between Alice and Bob, to avoid tampering. The newly generated secret is much longer than the initial MAC key. Unconditionally secure MACs exist, hence the whole scheme can be made unconditionally secure.}\]

4.4 Scenario 2: the satellite scenario

[Equations (4.8)–(4.11) are not part of the exam.]

The strategy

Beforehand, Alice and Bob agree an appropriate error-correcting code \(C\) with codewords of length \(N\). Alice chooses a random message \(R\), encodes it to \(V^N\) and sends \(V^N \oplus X^N\) to Bob over the noiseless public channel. Bob computes \(W^N := (V^N \oplus X^N) \oplus Y^N\). He accepts the received word only if \(W^N\) has much closer Hamming distance to some codeword in \(C\) than the error-correcting capability of the code. He tells Alice over the noiseless public channel if he accepts or not.

In this way Alice and Bob create a conceptual channel which is nearly noiseless for them, but noisy for Eve. Why is it noisy for Eve? By the removal of the unacceptable words, conditioning occurs on those events where Alice and Bob receive almost exactly the same noise \((X^N \approx Y^N)\) from the satellite. Eve’s noise, even for \(\varepsilon_E < \varepsilon_A, \varepsilon_B\), is independent. She has to guess at \(X^N\), and hence also the message \(R\).

Finally, Eve’s partial ignorance of \(R\) is leveraged by applying privacy amplification, e.g. by XOR-ing multiple messages together. (The piling-up lemma guarantees that Eve’s ignorance grows).

Example: repetition code

The repetition code of length \(N\) \((C_N)\) has only two codewords: \((0, 0, \cdots, 0)\) encoding the message ‘0’ and \((1, 1, \cdots, 1)\) encoding the message ‘1’. Alice generates a random \(R \in \{0, 1\}\) and sends \(A := (R, R, \cdots, R) \oplus X^N\) over the noiseless public channel. Bob accepts only if he exactly receives a codeword over the conceptual channel, i.e. if \(A \oplus Y^N \in C_N\). He tells Alice over the noiseless public channel if he accepts or not. The probabilities of Bob accepting and of accepting correctly/incorrectly are

\[p_{ok} = (1 - \varepsilon_A * \varepsilon_B)^N, \quad p_{error} = (\varepsilon_A * \varepsilon_B)^N, \quad p_{accept} = p_{ok} + p_{error}.\]

(4.6)
The new channel from Alice to Bob has an error rate

\[ \beta = \frac{p_{\text{error}}}{p_{\text{accept}}} \]

We introduce the notation \( \alpha_{bw} \) for the probability that a single bit 0 sent by Alice is received as \( b \) by Bob and as \( e \) by Eve.

\[
\begin{align*}
\alpha_{00} &= (1 - \varepsilon_A)(1 - \varepsilon_B)(1 - \varepsilon_E) + \varepsilon_A \varepsilon_B \varepsilon_E, \\
\alpha_{01} &= (1 - \varepsilon_A)(1 - \varepsilon_B)\varepsilon_E + \varepsilon_A \varepsilon_B (1 - \varepsilon_E), \\
\alpha_{10} &= (1 - \varepsilon_B)\varepsilon_B (1 - \varepsilon_E) + \varepsilon_A (1 - \varepsilon_B)\varepsilon_E, \\
\alpha_{00} &= (1 - \varepsilon_A)\varepsilon_B \varepsilon_E + \varepsilon_A (1 - \varepsilon_B) (1 - \varepsilon_E). 
\end{align*}
\]

We define \( q_w \) as the probability that the codeword \((0, \cdots, 0)\) sent by Alice is accepted by Bob (correctly or incorrectly) and received by Eve as a specific given word with Hamming weight \( w \),

\[ q_w = \alpha_{00}^{w} \alpha_{01}^{N-w} + \alpha_{10}^{w} \alpha_{11}^{N-w}. \]

(The first term has all 0s received by Bob, in the second term he receives all 1s.) Eve uses majority voting to guess \( R \). Conditioned on Bob accepting, her probability of wrongly guessing \( R \) is (for odd \( N \))

\[ \gamma = \sum_{w = [N/2]}^{N} \binom{N}{w} \frac{q_w}{p_{\text{accept}}}. \]

So the new channel has error rate \( \beta \) for Bob and \( \gamma \) for Eve. \( N \) is chosen such that \( \gamma > \beta \).

For the final step, privacy amplification, it is important to know how much information Eve has about \( R \). When Eve receives a word with Hamming weight \( w \), for her the probability of \( R = 0 \), given all her knowledge, is \( q_w/(q_w + q_{N-w}) \). Here \( q_w + q_{N-w} \) is the total probability that Hamming weight \( w \) occurs, namely starting from the 0-codeword or from the 1-codeword. For given \( w \) her ignorance about \( R \) is \( H(R | \text{weight} = w) = h(q_w/(q_w + q_{N-w})) \). Averaged over \( w \) this yields the following information that Eve has about \( R \),

\[ \mathbf{I}(R; Z^N, X^N \oplus e(R)) = \sum_{w = 0}^{N} \binom{N}{w} \frac{q_w}{p_{\text{accept}}} \cdot \left[ 1 - h \left( \frac{q_w}{q_w + q_{N-w}} \right) \right]. \]

Figure 4.3: Error rates \( \beta \) (lower curve) and \( \gamma \) (upper) as a function of \( N \), for \( \varepsilon_A = \varepsilon_B = 0.2 \) and \( \varepsilon_E = 0.15 \).
Figure 4.4: Bob's knowledge $1 - h(\beta)$ (upper curve) and Eve's knowledge (4.11) (lower curve) as a function of $N$ for $\varepsilon_A = \varepsilon_B = 0.2$ and $\varepsilon_E = 0.15$
Chapter 5

Physical Unclonable Functions (PUFs)

5.1 An Introduction to PUFs

Many physical objects have features that are unique and difficult to clone, and can therefore be used for identification in a way similar to biometrics. Some of these objects behave in a complex way when a stimulus is applied to them. The response of the object can be interpreted as the result of evaluating a parametrized function with the stimulus being the argument. The unique features of the object can be considered to be the (hidden) function parameters. Because of the similarity with mathematical functions, these unique unclonable objects are often referred to as Physical Unclonable Functions (PUFs), but terminology like Physical Random Functions or Physical One-Way Functions is used as well. Because of the cryptographic setting in which PUFs are used, stimulus and response are usually called a Challenge-Response Pair (CRP).

Although there is a formal definition of a PUF, the word is used rather loosely in the literature. The definition lists a number of requirements that have to be satisfied for an object to be called a PUF. However, the name is often also applied to physical structures that only partially meet these criteria.

5.1.1 History and concepts

In the history of cryptography there are many examples of physical objects used as aids in the encryption and decryption process. For instance, in the 5th century BC the Spartans used a device called the scytale, a cylinder of a certain radius. An instance of this scytale was owned by both sender and receiver. When encoding a message, the sender wrapped a narrow strip of cloth around the cylinder and wrote a message across it parallel to the axis of the cylinder. He then sent the unwound cloth to the receiver. Anyone intercepting the cloth sees a random-looking sequence of letters. When obtaining the cloth, the receiver would wind the cloth onto his own cylinder, thus decrypting the message.

Another example is the Cardan grille from the 16th century, a rectangle with a number of randomly placed holes in it. In order to encrypt a secret message, the sender wrote his message onto paper through the holes, and the remaining blank space on the paper was filled in, to form an inconspicuous fake message. The receiver owned the same grill and by putting it on top of the received message he could decrypt the message.

A more recent example is the Enigma cipher machine, an electro-mechanical device resembling a typewriter. It contained a.o. a typewriter keyboard, a number of mechanical rotors that electrically permuted the alphabet in a parametrisable way, a switchboard with 26 plugs for exchanging pairs of letters using cables, and as output it had lamps, one for each letter of the alphabet. The sender typed his plaintext, and the machine showed the ciphertext by switching on the right lamp. The
receiving party needed to have the same type of Enigma machine and to prepare it in exactly the same starting position as the sender. The process of decryption was exactly the same as encryption.

Clearly this list of examples is not complete, but it illustrates that physical objects have been used in a cryptographic setting for a long time. However, it is important to note that the physical objects in these examples are fundamentally different from PUFs in the sense that they have no unclonable features. They are merely (electro-)mechanical implementations of a mathematical algorithm, much like current-day computers running RSA or AES algorithms.

An important step in the thinking about unclonability in the context of cryptography was made by Wiesner in the late 1960s. He proposed an anti-counterfeiting method that makes use of the no-cloning theorem of quantum physics, which states that it is impossible to duplicate an unknown quantum state with a high probability of success. By equipping an item (e.g. a banknote) with a quantum system in a quantum state known only to the issuer, it is ensured that counterfeiters cannot clone the item, while the issuer is still able to verify its authenticity. This method is extremely secure but unfortunately, it is not very practical because quantum states are very difficult to maintain over long time spans. Therefore it is not to be expected that devices in the near future will contain quantum systems for security purposes. Consequently, in this book we concentrate on PUFs based on classical physics.

In 1991, Simmons proposed an anti-counterfeiting concept that was very different from the ideas at the time. Traditionally, authenticity marks that were embedded into items to protect them against counterfeiting, were all identical and difficult to forge. As Simmons noted, the requirement that all marks have to be equal, fundamentally contradicts the requirement that they have to be hard to forge and one should use unique, irreproducible physical features instead.

In 2001, Pappu [23] introduced the concept of a Physical Unclonable Function (PUF) or Physical One-Way Function (POWF). He proposed to use a PUF in a cryptographic challenge-response setting and defined a PUF as a physical object with the following properties:

1. The object can be subjected to a large number of different challenges that yield an unpredictable response.

2. The object is very hard to clone physically.

3. Mathematical modelling of the challenge-response physics is very difficult.

4. It is hard to characterize the physical structure of the object in a non-destructive way. (‘Opaqueness’.)

Property 1 ensures that there is enough randomness such that responses cannot be predicted. Property 2 refers to physical unclonability, whereas property 3 represents mathematical unclonability. The latter prevents an attacker using a model to mimic the behaviour of a PUF that is not physically present. Finally, property 4 hinders an attacker in gathering the data necessary to make a clone. It can be considered as an integral part of properties 2 and 3, but it is listed separately for clarity.

In order to satisfy all these properties, the object constituting a PUF needs to be highly complex. For property 2 the production process must be fundamentally uncontrollable. Furthermore, for properties 1 and 3 the responses need to be sensitive to small changes in the object’s structure and in the input. For property 4 the object needs to be ‘opaque’, i.e. hard to scrutinize by non-destructive measurement methods. This kind of overall complex behaviour is typically observed in manufacturing and measuring a large number of very small structures. In the following section, we give some practical realisations of PUFs.
5.1. AN INTRODUCTION TO PUFS

5.1.2 A short overview of PUF realisations

From the previous section it can be seen that physical structures with many small structures with random properties might serve as a PUF. Below, we list a number of PUF realisations and PUF-like structures. This list is not exhaustive, but illustrates the broad range of possible physical implementations.

In [25], the author describes a tamper-evident Integrated Circuit (IC) with a unique identifier or key based on an 'active coating'. The IC is covered with an opaque layer that contains a random mixture of conductors and insulators. Sensors inside the chip probe the coating and convert the measurement into a binary string. If the IC detects a change of the binary string it will take appropriate countermeasures. This basic concept was further developed in 2005 by Tuyls et al. (e.g. [32]), who coined the term coating PUF and showed explicitly how a key can be derived from measurements of the coating and how to use the result in a setting of secure key storage.

In 2001, Pappu et al. [23, 24] proposed to use three-dimensional optical structures as PUFs. These consist of a transparent medium containing many randomly positioned scatterers. When probed with a laser, the reflected or transmitted light is the result of multiple coherent scattering and forms a random-looking pattern of bright and dark areas known as speckle. The properties of the incoming laser beam, e.g. angle of incidence and focal distance, represent the challenge and the speckle pattern is the response. Such an Optical PUF system satisfies all four requirements.

In 2002, Gassend et al. [11] introduced the silicon PUF which are based on the fact that during manufacturing there are always small variations, even between ICs from the same wafer. While these variations do not harm the proper operation of ICs, they can be used as a source of randomness. In Silicon PUFs, the challenge is a pulsed time signal used as an input to a certain part of an IC, and the response consists of the delay times in the various wires and logic devices on the IC.

Another example is the SRAM PUF, introduced in 2007 [15]. Similar to a Silicon PUF, it is based on random variations during IC manufacture as follows. When a static RAM (SRAM) is switched on, the memory cells have an undefined state. When read out, a cell yields a ‘0’ or a ‘1’ depending on the precise characteristics of the cell, which are subject to random manufacturing inaccuracies. Hence, a freshly switched on SRAM can be challenged by choosing certain memory addresses, and the response is given by the returned start-up values.

In 2004, Kirovski [18] described a system based on a layer of randomly positioned glass fibers. The challenge is an ordinary beam of light illuminating a part of the layer. The response is a random-looking collection of fiber endpoints that light up.

In 2006, DeJean et al. [8] introduced so-called Radio-Frequent (RF) Certificates of Authenticity for anti-counterfeiting purposes. A random structure, containing conductive and dielectric parts, is probed by near-field RF waves over a range of frequencies, typically up to several gigahertz. The response is a set of frequency-dependent scattering parameters obtained from multiple antennas. Clearly there are many more physical structures that can be used as a PUF and an optimal choice will depend on the application in which a PUF will be used. Therefore, in the next section some application and use models will be discussed.
5.1.3 Applications and use models

The specific properties of PUFs, such as uniqueness and unclonability, make them useful for a large number of security applications. In a general setting, the presence of the correct PUF is verified by measuring the response to a challenge using a measurement device (reader) and comparing the actual response to a reference response which is stored in the verification system.

The actual PUF can be used in two main settings and we distinguish between uncontrolled or bare PUFs, and Controlled PUFs (CPUFs).

In the setting of an uncontrolled PUF, a reader interacts directly with the physical PUF structure. This setting is well suited for verification using trusted readers. Since a verifier knows that a trusted reader is scanning a physical object and is not otherwise tampered with, the only relevant property of the PUF is physical unclonability. Typical application examples are token authentication, anti-counterfeiting, copy protection, brand protection and tamper evidence.

In contrast, an untrusted reader cannot verify if it is interfacing with a real physical device. Therefore, besides unclonability, it should be infeasible to predict responses which can be achieved by choosing a PUF that contains much randomness and/or responds slowly to challenges. Possible application examples are remote authentication where the reader is under the control of the prover (e.g. on-line banking).

The concept of a controlled PUF was introduced in 2002 by Gassend et al. [12, 14, 13]. As opposed to the uncontrolled PUF, in case of a CPUF a reader interacts with the physical PUF through a control layer. The PUF and the control layer are inseparably bound together and any attempt to force the components apart will damage the PUF. As a result, an attacker has no direct access to the PUF because the control layer completely shields the PUF’s inputs and outputs. This extension to a physical PUF substantially strengthens the security, since an attacker cannot probe the PUF at will and cannot interpret the responses.

By further extending the functionality of the control layer, CPUFs enable a number of applications that are completely different from unprotected PUFs. One of these is secure key storage where the control layer derives a secret from the PUF and runs a strong authentication algorithm, e.g. a zero knowledge protocol to prove knowledge of the secret extracted from the PUF. In this setting a very limited number of CRPs is sufficient, and one speaks of a Physically Obscured Key (POK) instead of a PUF.

Incorporating even more functionality in the control layer allows for two special applications: certified execution and certified measurement. In certified execution, a user outsources a computation task to the control layer of a CPUF. The control layer performs the computation and provides a proof that the job was run on that particular machine. In certified measurement, the CPUF contains an additional integrated sensor such as a microphone or video camera. All data recorded by the sensor is certified by the CPUF, for example by creating a signature using a PUF key as a private key. This allows for verifying both the integrity and the origin of the data.

5.1.4 Comparison with biometrics

On the one hand, PUFs are very similar to biometrics concerning issues such as feature extraction, analog to digital conversion, measurement noise and template protection. On the other hand, there are important differences. Perhaps the most important difference, from a security point of view, is that it is much easier to keep PUF responses secret than biometric data. A PUF can be embedded in an electronic device that controls access to the PUF, but a biometric feature of the body is always vulnerable to surreptitious measurements by attackers. It makes perfect sense to derive secret keys from CRPs, whereas doing such a thing with biometric data is perilous. Furthermore, if biometric secrets are stolen, one cannot re-issue a new body part; re-issuing a new PUF, on the other hand, is feasible. For these reasons, PUFs can be used in many more (cryptographic) applications than biometrics. Another important difference is that, obviously, a person always has direct access to his biometric, whereas a PUF can be left behind, intentionally or unintentionally.
5.1.5 Current directions of PUF research

When building practical systems based on PUFs, apart from the important intrinsic properties of the PUF, several other requirements must be met. Like in biometrics, a PUF system contains the step of measuring the properties of a physical entity and this will give rise to measurement noise. Because cryptographic primitives such as hashing, signing, and encryption by definition amplify any noise in their input, dedicated measurement, signal processing and quantization techniques must be developed, possibly in combination with error correction, to eliminate the noise. However, this introduces a new problem because side-information (redundancy data) needs to be stored outside the PUF in insecure storage, where it potentially leaks information about the PUF responses. Many of these issues are similar to the situation where noisy biometrics are combined with cryptography and must be taken into account when designing PUF-based security systems. (An important difference between PUFs and biometrics, however, is that with PUFs one has some freedom to design physical structures that carry more information than the biometric modalities.)

The problem of how to do error correction combined with non-leaking redundancy data is a recent research topic called ‘Fuzzy Extraction’. We will see more of this in Chapter 6.

More on the practical side, for mass deployment (e.g. in the case of anti-counterfeiting) PUFs must be cheap to produce and it must be possible to use them in combination with low-cost readers. Furthermore, since PUFs are based on physical systems, the security depends on the sophistication and resources of the attacker. This is similar to side-channel attacks against physical implementations of traditional cryptographic systems. These problems with engineering attacks must be studied thoroughly before a PUF system can be successfully deployed.

One of the trends in chip design is the integration of ever more functionality into one piece of silicon. This means that the cheapest kind of PUF would be one that is embedded in the small-scale structure of chips. The trick is to find out the best way to do this using the properties of available components, such as memory start-up values, component delays, remanence effects etc.

One of the latest developments is to combine the unclonability of (classical) PUFs with the unclonability of quantum states. This leads to unusual security primitives such as secure measurement without trusted devices, and authenticated quantum channels.

5.2 A very quick introduction to Fuzzy Extractors

Consider an enrollment measurement of a PUF, yielding result \( X \), and a later measurement of the same PUF (during authentication) yielding \( X' \). Due to noise, \( X' \) is not necessarily equal to \( X \), but it is expected to be close. A Fuzzy Extractor consists of two algorithms, \( \text{Gen} \) and \( \text{Rep} \). The \( \text{Gen} \) algorithm is applied during enrollment; from \( X \) it creates a secret \( S \) and enrollment data \( W \) (also known as helper data or public data). The \( W \) is allowed to be publicly known. Ideally, \( W \) reveals no information about \( S \). The \( W \) serves as error correction data (redundancy data) for the authentication phase. The \( \text{Rep} \) algorithm takes \( X' \) and \( W \) as input and from them tries to reconstruct the secret \( S \). If the noise is not larger than anticipated, then \( \text{Rep} \) succeeds.

A Fuzzy Extractor combines error correction with secrecy of the key extracted from \( X \). This is a nontrivial combination of requirements. We will investigate Fuzzy Extractors in proper detail in Chapter 6.
5.3 Examples of PUFs

5.3.1 Optical PUFs

At the moment only one physical system is known that satisfies all four requirements in the definition of a PUF (Section 5.1.1): the optical PUF originally proposed by Ravikanth Pappu in 2001. Multiple scattering of laser light in a disordered three-dimensional medium is a very complicated process.

- The resulting speckle pattern strongly depends on the wavelength, angle and focus of the laser beam. There are many different input configurations; due to the complexity of the scattering they lead to output that looks random.

- Making a physical clone (given exact knowledge of the PUF’s internal structure) requires a highly controlled process to put small particles in exactly the right position in a three-dimensional medium. This is difficult and costly (though of course not impossible in principle).

- It turns out that mathematical modeling of multiple scattering is notoriously difficult. Even for microwaves it is near impossible make accurate predictions, let alone for optical wavelengths. Hence, even an attacker who knows all the details of an optical PUF’s internal structure cannot reliably compute responses.

- An interesting side effect of scattering is that it is difficult to look deeply into a PUF. The diffusion of light hides the exact positions of the scattering particles. It is currently estimated that you can look twenty scattering-lengths into a medium using various advanced microscopy techniques. One has to be careful with this number, however, e.g. in view of recent advances in ‘inverting diffusion’ [34].

Translating terms like ‘difficult’ and ‘costly’ into more numerical statements is not easy. Manufacturing techniques change over time. We have more success in translating phrases like ‘random looking’ and ‘many challenges’ into numbers. Optical PUFs can be effectively described using the wave guide model.

Wave guide model

Consider a little slab of thickness $d$ in the $z$-direction and cross-section $A = W^2$ in the $(x, y)$-plane. The laser light has wavelength $\lambda$. When light travels through the slab, it goes a certain distance $\ell$ (on average) before scattering; this length is called the scattering length. We have $\lambda \ll \ell \ll d$.

The slab can be seen as a ‘wave guide’ letting light waves travel in the $z$-direction. The waves, which also have some velocity in the $(x, y)$-plane, bounce around in the guide, reflected by the ‘walls’. A certain value of the $(x, y)$-velocity is called a mode. The waveguide allows a number of transversal modes $N_{\text{mod}}$. The scattering process is represented by an $N_{\text{mod}} \times N_{\text{mod}}$ random scattering matrix $S_{ab}$, specifying how much light is scattered from incoming mode $b$ to outgoing mode $a$.

![Wave guide model diagram](image)

How many modes are there? $N_{\text{mod}}$ is determined by the wavelength and the geometry of the slab. Instead of the wavelength $\lambda$ it is often useful to work with its inverse,

$$k = \frac{2\pi}{\lambda},$$

(5.1)
5.3. EXAMPLES OF PUFS

called wave number. The energy of a photon with wave number \( k \) is given by \( c \sim k \), with \( c \) the speed of light and \( h = h/(2\pi) \), where \( h \) is Planck’s constant. Since light has a direction, we can define a wave vector \( \mathbf{k} \) of length \( k \), and pointing in the traveling direction. The electric field (in complex notation \( E_x + iE_y \)) at the PUF surface can be represented as a Fourier sum. Since the field is confined on a square area of size \( W \times W \), the allowed wave vectors are

\[
(k_x, k_y) = \frac{2\pi}{W}(a_x, a_y)
\]  

(5.2)

with \( a_x, a_y \in \mathbb{Z} \). The Fourier transform is given by

\[
E(x, y) = \sum_{a_x,a_y \in \mathbb{Z}} C(a_x, a_y)e^{i(xk_x+yk_y)},
\]

(5.3)

Conservation of photon energy constrains the summation to only those \( a_x, a_y \) that satisfy \( k_x^2 + k_y^2 \leq k^2 \). So the electric field, which lives on a continuum, is completely described by a finite number of parameters \( C(a_x, a_y) \). Each allowed momentum value \((k_x, k_y)\) in the Fourier sum is called a mode. The total\(^1\) number of modes is given by

\[
N_{\text{mod}} = \# \{(a_x, a_y) \text{ with } k_x^2 + k_y^2 \leq k^2\} = \frac{\pi A}{\lambda^2}.
\]

(5.4)

The integers \( a_x, a_y \) lie in the range \((-W/\lambda, W/\lambda)\).

**Example 5.1** Let’s now fill in some realistic numbers: \( W = 1.0 \, \text{mm} \), \( d = 1.0 \, \text{mm} \), \( \ell = 10 \, \mu\text{m} \) and \( \lambda = 500 \, \text{nm} \). This gives \( N_{\text{mod}} = 1.3 \cdot 10^7 \) transversal modes. This is the number of possible (physically independent) challenges, quite a large number!

How big is a speckle? The modes can be seen as independent degrees of freedom of light. That means that one mode may be ‘full’ and the next ‘empty’. So the size of a speckle is determined by the angular size of a mode. The angular separation \( \Delta \varphi \) between modes is estimated as follows. Consider the difference between the mode \((a_x, a_y) = (0, 0)\) and the mode \((1, 0)\). Since \( \Delta \varphi \) is small we are allowed to write \( \Delta \varphi \approx \tan \Delta \varphi \). We have

\[
\tan \Delta \varphi = \frac{k_x}{k} = \frac{2\pi/W}{2\pi/\lambda} = \frac{\lambda}{W}.
\]

(5.5)

So from all this physics we are able to draw some simple conclusions about speckles: they become smaller when you increase the spot size of the laser beam \((W^2)\) and when you decrease the wavelength \( \lambda \). Furthermore, they become bigger (in the sense of absolute size, not angles) when you increase the distance between the PUF and the detector.

Some of the light is transmitted through the PUF, the rest is reflected. (The model does not consider absorption.) The scattering process can be represented as a complex random matrix \( S \), whose elements map incoming states to outgoing states,

\[
\vec{E}_{\text{out}}^a = \sum_{b=1}^{N_{\text{mod}}} S_{ab} \vec{E}_{\text{in}}^b.
\]

(5.6)

However, light intensity (the thing that most detectors measure) is not given by the electric field but the field squared, \( E_x^2 + E_y^2 \). The transmission coefficients for the intensity are defined as \( T_{ab} = |S_{ab}|^2 \).

\(^1\)If polarisation is taken into account, the number of modes doubles. We will not consider polarisation.
CHAPTER 5. PHYSICAL UNCLONABLE FUNCTIONS (PUFS)

5.3.2 Coating PUFs

Coating PUFs are integrated with a chip (see Figs. 5.2 and 5.3). The Integrated Circuit (IC) is covered with a coating consisting of e.g. aluminophosphate, which is doped with random dielectric particles. By random dielectric particles we mean several kinds of particles of random size and shape with a relative dielectric constant $\varepsilon_r$ differing from the dielectric constant of the coating matrix. In order to challenge the coating PUF, an array of metal sensors (e.g. a comb structure of wires), is laid down directly beneath the passivation layer. Sufficient randomness is only obtained if the dielectric particles are approximately of the same size as the distance between the sensor parts, or smaller.

A challenge corresponds to a voltage of a certain frequency and amplitude applied to the sensors at a certain point of the sensor array. Because of the presence of the coating material with its random dielectric properties, the sensor plates with the material in between behave as a capacitor with a random capacitance value. The capacitance value is then converted into a bit string which can be used as an identifier or a key. It may happen that only a single key can be derived. Then it is more accurate to speak of a coating POK.

The coating needs to have a high average $\varepsilon_r$ in order to accommodate large variations. We briefly describe trials done at Philips Research [31], to show the orders of magnitude involved. In these experiments the coating matrix was made of Al(PO$_3$)$_3$, which has a dielectric constant of 7–8. Mixed through the matrix there were TiO$_2$ and TiN particles. The TiO$_2$ particles have a dielectric constant in the order of 70–180. The TiN particles are conductive; apart from causing large variations in the capacitance, it also shields off electromagnetic leakage from the IC. Furthermore, the TiN particles are mechanically tough and chemically resistant to many nasty substances. The coating was a few micrometers thick. The wires of the sensors were a few $\mu$m wide and tens of $\mu$m long. The capacitance measurement was based on a precise voltage comparison and a stable current source: The current source pumps charge into the sensor. When the voltage reaches a certain threshold $V_{\text{high}}$ the direction of pumping is reversed. Then when the voltage reaches a
5.3. EXAMPLES OF PUFS

**Figure 5.4:** Logical circuit representation of an SRAM cell.

second threshold $V_{low}$ the direction is reversed again, etc. This process leads to a fast oscillation of which the frequency can be measured accurately. It turned out that there was enough entropy in the coating to extract at least one AES key per mm$^2$ of surface area.

Formal properties
This kind of coating and readout achieves opaqueness and physical unclonability. The number of challenges is very limited, possibly only 1. The coating is probably easy to model mathematically. Coating PUFs seem to be a good candidate for achieving read-proof tamper-evident key storage (Section 5.4.4).

**Exercise 5.1** It was said above, without a full explanation, that “sufficient randomness is only obtained if the dielectric particles are approximately of the same size as the distance between the sensor parts, or smaller”. Why is this the case? What goes wrong if the dielectric particles are too small? And if they are too big?

5.3.3 SRAM PUFs

Static Random Access Memory or SRAM is a type of volatile memory abundantly used in digital devices. In its popular form, one SRAM cell occupies six transistors that constitute a logic circuit with two cross-coupled inverters, as shown in Fig. 5.4. Such a circuit can assume two stable logical states: $(A, B) = (0, 1)$ and $(A, B) = (1, 0)$, and hence stores one binary digit. External connections can inspect the state of the cell (read) or alter it (write).

When powered up, the behaviour of an SRAM cell is undefined. From Fig. 5.4 we expect it to be in a logically undetermined and unstable state right after power-up. Due to physical mismatch between the inverters and due to noise, the cell will quickly converge to one of the two stable states. Since both the inverters’ mismatch and the noise are governed by stochastic processes, the power-up state of an SRAM cell will be ‘random’. All the manufacturing inaccuracies are random, but fixed once the cell is manufactured. The thermal/power noise is different at every moment in time. Experiments [15] show that a certain fraction of cells has a tendency to often start up in a preferred state. The location of such ‘stable’ cells in a memory array varies from chip to chip and is unpredictable. Hence the stable cells can be used to obtain a unique fingerprint (or key) of the SRAM hardware.

Some care must be taken to avoid memory retention effects [16]. Many types of memory retain their state for a certain amount of time after having their power switched off. This may affect the startup values if the device is immediately switched on again. Experiments [15] have shown that retention effects in SRAM are negligible at normal temperatures.

Typically, the helper data will contain a set of pointers to stable cells. The fingerprint obtained from them is still a bit fuzzy because the cells are not perfectly stable: there is thermal noise and power fluctuations. If the objective is to obtain a perfectly reproducible bitstring (e.g. for use in a cryptographic algorithm), then additional helper data is necessary to do error correction.

Formal properties
There is only a single challenge. The SRAM is not opaque: anyone with access to it can measure its startup properties. The SRAM can be considered to be physically unclonable. Mathematical cloning is not hard.
5.3.4 Delay based PUFs

'Silicon PUFs', first proposed in [11], are based on uncontrollable process variations during IC manufacture. These variations lead to subtle but measurable differences in signal delay times. Across a chip, signal propagation delays vary due to mask variations; this is sometimes called the system component of delay variation. There are also random variations in chips across a wafer, and from wafer to wafer due to, for instance, process temperature and pressure variations during the various manufacturing steps. The magnitude of delay variation due to this random component can be 5% or more. There is a continuous effort to reduce all these sources of variation because they inherently limit the component density of the IC. Nevertheless, the relative variations in state of the art components tends to increase with shrinking device sizes.

Arbiter PUF

Fig. 5.5 shows a schematic overview of an ‘arbiter PUF’. There is a top path and a bottom path, each with their own delay buffers. A signal is simultaneously sent into both paths. In each segment $i \in \{1, \ldots, n\}$, the difference between the top delay and the bottom delay is denoted as $\Delta_i$. The challenge bits $c_i \in \{-1, +1\}$ are fed into switch blocks. At the $i$'th switch block the top and bottom signals cross if $c_i = -1$; otherwise the signals stay in their top/bottom path. The Arbiter detects which of the signals arrives first. If it is the top one, the Arbiter outputs the response $R = +1$; if it is the lower one, it outputs $R = -1$. Let $\vec{c}$ be a vector consisting of the challenge bits $c_i$, and let $\vec{\Delta}$ be a vector consisting of the delay differences $\Delta_i$. Then the response of the arbiter PUF is given by

$$R(\vec{c}) = \text{sgn}(\sum_{i=1}^{n} c_i \Delta_i) = \text{sgn}(\vec{c} \cdot \vec{\Delta}).$$

One aims to have a certain amount of crosstalk between the components, so that $\vec{\Delta}$ depends on $\vec{c}$. However, it turns out in practice that such dependence is minimal. We can think of the $\vec{\Delta}$ in (5.7) as a constant that depends on the PUF instance but not on the challenge.

There are $2^n$ possible challenges. However, because of the additive behaviour of the $\Delta$’s, it is relatively easy for an attacker to predict the response to a fresh random challenge after he has observed a small number of Challenge-Response Pairs.
In order to make the analysis of CRPs more difficult it has been proposed [21] to couple multiple Arbiter PUFs together. This is depicted in Fig. 5.6. All the PUFs receive the same challenge vector \( \vec{c} \); their outputs are multiplied to yield a single response \( R(\vec{c}) \in \{-1,1\} \). Let us denote the \( \Delta \)-vector of the \( j \)'th PUF as \( \vec{\Delta}^{(j)} \), and the response of the \( j \)'th PUF as \( R^{(j)}(\vec{c}) \). Then

\[
R(\vec{c}) = \prod_{j=1}^{k} R^{(j)}(\vec{c}) = \text{sgn}[\vec{c} \cdot \vec{\Delta}^{(1)} \cdot \vec{\Delta}^{(2)} \cdots \vec{\Delta}^{(k)}].
\] (5.8)

It is much harder to analyze CRPs that were generated by (5.8) than by (5.7). In spite of this fact, it was demonstrated in [9] that the system of combined Arbiter PUFs is vulnerable to a so-called Machine Learning attack. Such an attack is capable of estimating all the \( \Delta^{(j)} \) values from a limited number of CRPs. The required number of CRPs scales roughly as \( kn \). Thus we can conclude that even combined Arbiter PUFs do not support a large number of independent CRPs.

Oscillator PUF

An Oscillator PUF is very similar to an Arbiter PUF. Instead of an arbiter, there is a path selector which takes a binary challenge \( c_{n+1} \) as input. (Fig. 5.7.) The selector selects the upper branch if \( c_{n+1} = 1 \), otherwise the lower branch. The signal in the selected branch is fed back as input to the PUF. This feedback causes an oscillation whose period depends on the precise delays in the delay buffers. The oscillation period can then be transformed into a discrete response in several ways, e.g. by measuring the oscillation frequency and discretizing it.

The Oscillator PUF has the same vulnerability to Machine Learning attacks as Arbiter PUFs.

Formal properties of delay-based PUFs

The number of different challenges is exponential in \( n \). However, because of the Machine Learning attacks, the number of independent CRPs is linear in \( n \). If the IC blocks direct access to the delay measurements, then it is hard for an attacker to determine the delays; then the ‘opaqueness’ property holds. The uncontrollability of the manufacturing variations implies that physical cloning is difficult.

### 5.3.5 Comparison of PUF types

<table>
<thead>
<tr>
<th>Technology</th>
<th>many indep. CRPs</th>
<th>opaque</th>
<th>phys. unclonable</th>
<th>math. unclonable</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D optical</td>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>coating</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>SRAM</td>
<td>-</td>
<td>if access blocked</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>delay</td>
<td>-</td>
<td>if access blocked</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>
5.4 Detailed list of PUF applications

Here we give an overview of application scenarios for PUFs/POKs. We try to present this in a way that is as structured as possible, explicitly giving the attack model and the assumptions.

5.4.1 Anti-counterfeiting with bare PUFs

We consider one of the following scenarios

1. A bare PUF is created and embedded into a valuable product.
2. A valuable product is created. Some part of it is a bare PUF.

These scenarios are equivalent in the sense that they result in a product containing a bare PUF. Then the PUF is enrolled. This means that its characteristics are measured and stored for later reference. The storage has to be done in some trusted way, e.g., a trusted database or with a digital certificate signed by the enrollment authority.

At some later time, a verifier (Victor) wants to check if the product is authentic. He fetches the enrollment data for this specific object. (If the data comes accompanied with a certificate, he checks the certificate using the enrollment authority’s public key.) He measures the characteristics of the bare PUF and compares them to the enrollment data. If they are sufficiently close, he considers the product to be authentic.

The PUF has short-term validity. In the database case, it gets removed from the database after some time. In the certificate case, the certificate mentions an expiration date.

Attack model

We say that an attack is successful if an attacker can earn money in the following way. He manufactures a non-authentic product and embeds a self made bare PUF into it. The combination of counterfeit product, PUF (and possibly a fake certificate) has to pass Victor’s verification.

We do not consider attacks where an enrolled PUF is removed from an authentic product and put into a counterfeit product. With such a scheme the attacker cannot earn money. He either has to buy authentic products (too expensive) or to steal them (theft is a separate transgression in itself and out of scope of simple counterfeiting attacks).

Requirements for security

We need physical unclonability (property 2) and none of the other properties. It does not matter if there is only one challenge; or if the attacker knows everything about the PUF he wants to clone; or if he has a perfect mathematical model. The only thing that counts is that Victor does a measurement of a physical object, and the attacker has to make a passable physical clone of that object. (In fact the word PUF is not applicable at all. Unfortunately we are stuck with this name in the scientific literature at the moment.)

What can we say about the difficulty of physical cloning? Very little! The only generic theoretical analysis we have is based on the ‘birthday collision’ attack. In this attack, the attacker knows the properties of $M$ enrolled PUFs. He manufactures $N$ PUFs at random and hopes that one of them matches one of the $M$ known PUFs. Let’s say for simplicity that the PUF creation process has $b$ bits of entropy, and that there are $2^b$ possible distinguishable PUFs. When one PUF is created, the probability of hitting one specific PUF is $2^{-b}$. The expected number of successfully cloned PUFs is

$$\hat{N}_{\text{clones}} = N M 2^{-b}.$$  \hfill (5.9)

Let us denote the cost of creating a random PUF as $c$. Let the profit derived from a single cloned PUF be $s$. The attack is too expensive (on average) if $N c \geq \hat{N}_{\text{clones}} s$. Rewriting this condition in terms of $b$ we get

$$b \geq \log(M s c^{-1}).$$  \hfill (5.10)

**Example 5.2** If it is reasonable to assume that (i) only random PUF production is feasible as an attack; (ii) the attacker has full knowledge of no more than $M = 100$ PUFs, (iii) $s/c = 1000$, then 17 bits of PUF entropy is sufficient for successful anti-counterfeiting.
5.4.2 Remote authentication of a bare PUF

Consider a scenario where Alice wishes to check remotely if Bob has access to one specific enrolled PUF that is known to Alice. Here ‘access’ is defined very broadly, allowing Bob to relay Alice’s challenges to someone who has actual access to the PUF. Bob (or an attacker) claims that he has the PUF; Alice then starts an authentication protocol.

**Attack model**

Alice and Bob communicate over an insecure channel. The attacker does not know the enrollment data. At some points in time, we allow the attacker an opportunity to examine the PUF for a ‘reasonable’ amount of time, without Alice or Bob noticing this. This may happen multiple times, even between authentications. At the moment when Alice requires authentication, the attacker does not have access to the PUF. The attack succeeds if the attacker convinces Alice that he *does* have access.

**The bare PUF authentication protocol**

At enrollment, a number of randomly selected challenges is fed to the PUF. The CRPs are securely stored in Alice’s database. The PUF is handed to Bob. The authentication protocol consists of the following steps:

1. Alice selects a random challenge from the database. She sends it to Bob.
2. Bob feeds the challenge to the PUF and measures the (analog) response. He sends this response to Alice in the clear.
3. If the received response is sufficiently close to the enrolled value, then Alice is convinced that her communication partner (whoever that may be) has access to the PUF; in that case she flags the used up CRP so that it will never be used again.

**Requirements for security**

The protocol leans very heavily on the quality of the PUF. The following properties must hold:

A. The number of independent CRPs must be so large that the attacker’s ‘reasonable’ amount of time is too short to learn enough about the PUF.

B. At least one of the following:
   1. Opaqueness.
   2. Physical and mathematical unclonability.

If property A does not hold, then the attacker can learn a large fraction of all possible CRPs, and has a good chance of knowing the answer to Alice’s challenge. Property B1 prevents the attacker from characterizing the physical structure of the PUF; not knowing the structure, he cannot even attempt to clone the PUF in any way. If the PUF structure can easily be characterized, then property B2 makes sure that this knowledge cannot be exploited.
5.4.3 Remote authentication of a bare PUF, plus authenticated channel

![Figure 5.8: Example protocol for one-directional authentication and MAC key establishment based on a bare PUF.]

We consider a scenario that is almost equal to Section 5.4.2, but with one addition: Alice and Bob wish to derive a MAC\(^2\) key from the PUF response, so that the channel from Bob to Alice is turned into an (insecure) authenticated channel\(^3\)

**Attack model:**
Same as in the previous section.

**Protocol for remote bare PUF authentication and key agreement**

At enrollment, randomly selected challenges \(\{C_i\}_{i=1}^n\) are fed to the PUF. The \texttt{Gen} algorithm of a fuzzy extractor is applied to each measurement result \(X_i\), yielding secrets \(S_i\) and helper data \(W_i\). The triplets \(\{C_i, W_i, S_i\}_{i=1}^n\) are securely stored in Alice’s database. The PUF is handed to Bob.

The authentication protocol consists of the following steps:

1. Alice selects a random challenge \(i\) from the database. She generates a nonce \(u\). She computes an encryption \(C_A = E(S_i, u||f(u))\) using key \(S_i\); here \(f\) is a hash function. She sends to Bob: \(C_i, W_i, C_A\).

2. Bob feeds the received challenge to his PUF. He applies the \texttt{Rep} algorithm to his measurement outcome \(X'\) and the received \(W_i\), obtaining a key \(S'\). He computes a decryption \(D(S', C_A)\) which he then parses as \(u'||a\). He checks if \(a = f(u')\). If this is false, he aborts the protocol. If it is true, then he trusts that he is talking to someone who knows the PUF response to \(C_i\).

   Bob computes an encryption \(C_B = E(S', u')\). He sends \(C_B\) to Alice.

3. Alice checks if \(D(S_i, C_B) = u\). If this is false, the authentication has failed. If true, then Alice trusts that she is talking to someone who knows \(S_i\) (which should imply it’s the PUF owner); she flags the entry \(i\) in her database so that it will not be used again.

4. Bob uses \(S_i\) as a MAC key.

[Many variants can be constructed of the above protocol. The upshot is that Alice is convinced she’s talking to someone who has access to the PUF and Bob is convinced he’s talking to someone who knows a CRP of his PUF. The protocol hides the MAC key from eavesdroppers.]

Note that in our attack model it is easy for an attacker to obtain random CRPs of Bob’s PUF. He can use these CRPs to impersonate Alice. Hence, the channel from Alice to Bob is not authenticated.

\(^2\)MAC = Message Authentication Code

\(^3\)They could of course create such a channel based on an a priori shared secret, i.e. something that Bob knows. However, the protocol involving the PUF achieves something different: Alice verifies if Bob possesses something unique and unclonable.
5.4. DETAILED LIST OF PUF APPLICATIONS

Requirements for security
Same as in the previous section. Apart from that, a Fuzzy Extractor is needed.

5.4.4 Read-proof, tamper evident storage of keys

Many devices need to store cryptographic keys in a secure way. (Smartcards, cell phones, authentication tokens, DVD players, payTV decoders, access passes, etc.) The security must be maintained under extremely hostile conditions. Here we consider a key storage method based on a PUF that needs no more than a single challenge. Such a system is known as a Physically Obscured Key (POK).

Attack model
The attacker has unlimited access to the device. He is allowed to do anything to the device. We consider the attack successful if he learns the stored key. In addition, we also count it as success if the attacker modifies the stored key (even without knowing the key) in any way and the device does not notice the change.

Note: side channel attacks are not considered here. They form a class of attacks that is unrelated to the storage mechanism.

Key storage method
A random PUF is created. It is joined inseparably to a readout mechanism and processor in an electronic device. At enrollment the device measures the PUF output $X$. The $\text{Gen}$ procedure is applied to $X$, resulting in a secret $S$ and helper data $W$. The device’s sensitive data (such as device keys etc.) are encrypted with $S$; the ciphertext is stored in insecure memory visible to the attacker. A cryptographic “certificate” $Q_W$ is computed, for safeguarding the authenticity of $W$. (We are deliberately vague about this step at the moment. Later we will see several ways of implementing this.) The $W$ and $Q_W$ are stored in insecure memory as well.

Later, when the device has to access its sensitive data, it performs the following steps. It reads $W'$ and $Q_{W'}$ from the insecure memory. (These values may have been manipulated by the attacker.) It challenges the PUF and measures output $X'$. It computes $S' = \text{Rep}(X', W')$. It checks a cryptographic consistency relation\(^4\) between $S'$, $W'$ and $Q_{W'}$. If the consistency check fails, the device considers itself under attack and does not proceed. In case of consistency the device knows that $S' = S$. It decrypts the ciphertext in the insecure memory using the key $S'$.

Requirements for security
The following properties must hold:

A. The connection between the PUF and the device electronics has to be so intricate that any attempt to separate the components from each other damages the PUF. Here ‘damage’ means that the PUF output changes significantly.

B. At least one of the following:
   1. Opaqueness. (In case of a nondestructive attack.)
   2. Physical and mathematical unclonability. (In case the PUF structure is determined by destructive attack.)

Property A protects the system against attacks where the PUF is detached from the device and coupled to a hostile device. Together with the cryptographic consistency check this achieves tamper evidence.

5.4.5 Generic anti-counterfeiting using POK’ed RFID tags

We sketch how POKs in RFID tags can be used for a generic anti-counterfeiting scheme. We will assume that there is a Public Key Infrastructure. There is a Certification Authority (CA) whose public key $P_{CA}$ is known worldwide and which is contained in the ROM of all RFID tags.

\(^4\)In fact, if there is a Public Key Infrastructure then consistency between $W'$ and $Q_{W'}$ suffices, and can be checked even before the PUF measurement is done.
Attack model
An attacker is considered successful if he can earn money by creating or reconfiguring unauthentic RFID tags in such a way that they pass the authentication protocol.
We emphatically do not consider the kind of attack where the tag is removed from an authentic product and put on a counterfeit product. Here the attacker either has to buy an authentic product or to steal one. Theft is out of scope. The only way to make money if he first buys authentic goods is to sell them on the black market; but (a) he could then just as well have sold the counterfeit goods on the black market, and (b) the existence of a black market where no-one cares about authenticity is out of scope.

Enrollment
1. An RFID tag is created with a random POK. In its ROM it has identifier $i$. The tag queries the POK and gets (analog) response $X$. It computes $(S, W) = \text{Gen}(X)$. The key $S$ is treated as a private asymmetric key. The tag computes a public key $P$ from $S$. The tag outputs $i$, $W$ and $P$.
2. The CA decides to couple the tag to a product with descriptor $m$. (The descriptor may comprise product type, product identifiers, URLs, time stamp, expiration date, etc.) The CA creates signatures $\sigma_W$ and $\sigma_P$ using its private key $S_{\text{CA}}$: $\sigma_W = \text{Sign}(S_{\text{CA}}, i||W)$ and $\sigma_P = \text{Sign}(S_{\text{CA}}, i||P||m)$. The CA gives $\sigma_W, \sigma_P, m$ to the tag.
3. The tag verifies the signatures using $P_{\text{CA}}$, i.e. it does consistency checks $\text{Verify}(P_{\text{CA}}, i||W, \sigma_W)$ and $\text{Verify}(P_{\text{CA}}, i||P||m, \sigma_P)$. If both signatures pass the test, the tag stores $W, \sigma_W, P, m, \sigma_P$ in its flash memory, which is insecure. Otherwise it generates an error message.
4. If there is no error message, the CA attaches the RFID tag to the product.

Authentication
1. A consumer inspects the product and its associated tag. His reader queries the tag.
2. The tag powers up in the field of the reader. The tag accesses its ROM to retrieve $i, P_{\text{CA}}$ and its flash memory to retrieve $W, P, m, \sigma_W, \sigma_P$. Then it runs $\text{Verify}(P_{\text{CA}}, i||W, \sigma_W)$ and $\text{Verify}(P_{\text{CA}}, i||P||m, \sigma_P)$. It only proceeds if both checks are passed. The tag does a PUF measurement, obtaining $X'$. It computes $S' = \text{Rep}(X', W)$. From $S'$ it computes a public key $P'$. The tag only proceeds if $P' == P$. The tag sends $i, P, m, \sigma_P$ to the reader.
3. The reader runs $\text{Verify}(P_{\text{CA}}, i||P||m, \sigma_P)$. Optionally, the reader also checks if $i, m$ is consistent with information about the product that comes from other sources. If everything is all right, the reader proceeds to the following step.
4. The reader and the tag run an authentication protocol in which the tag has to prove that it knows the private key $S$ associated with $P$. (E.g. a Zero Knowledge protocol.) If the tag passes this test, the reader is convinced that the tag is genuine.

Security analysis
There are several lines of attack. (i) Re-use existing signatures, putting them into cloned tags; this is infeasible because the tags have a POK. (ii) Manufacture tags with random POKs, then create signatures $\sigma_W, \sigma_P$ for those POKs. This is infeasible because of the cryptographic hardness of forging a digital signature. (iii) Deduce $S$ by eavesdropping on the authentication protocol, then create a POK-less tag that has $S$ in its memory. This is infeasible because of the strong crypto in the authentication protocol. (In the case of a ZK protocol it is even provable that eavesdropping yields exactly zero bits of information about $S$.)
The scheme has several beautiful properties. The private POK key $S$ never leaves the tag. Furthermore, the tag readers do not need access to any secret. Sooner or later, readers will always be hacked; in this scheme, such a hack causes no security danger.

5If the attacker can modify the tag’s ROM, he may replace the CA public key. Hence the tag may be fooled into accepting fake signatures. However, the reader has access to the true CA public key and is not fooled.
5.4. DETAILED LIST OF PUF APPLICATIONS

Practicality
The scheme requires public key operations on a cheap tag. Is this realistic? It seems to be on the edge. There are known implementations of Elliptic Curve Cryptography (ECC) in 13,000 gates, which is comparable to hash functions.

Exercise 5.2 An alternative verification method would be to combine $\sigma_W$ and $\sigma_P$ into a single signature on the concatenation $W||P$. This would also detect tampering with the stored $W$ and/or $P$, and it would require less computation by the tag. Can you think of a reason why it is better to first verify the correctness of $W$ and then of $P$?

Exercise 5.3 Consider an anti-counterfeiting tag without public key crypto. During enrollment it communicates a secret symmetric key $S$ to the CA. This happens in a secure environment. Can you think of security risks in this scheme that do not exist in the asymmetric case?

5.4.6 A POK without Public Key Infrastructure

It is instructive to show how the helper data $W$ can be protected from tampering when there is no certification infrastructure available to sign $W$. We consider a device that has to protect its device secrets, without the need to communicate to the outside world.

Method based on hash function

Initialization
The device does a PUF measurement, obtaining $X$. It computes $(S, W) = \text{Gen}(X)$. It stores $W$ and $\sigma = f(S||W)$ in its insecure flash memory. ($f$ is a hash function.) In the future, $S$ will be used as a key to encrypt/decrypt device secrets stored in the insecure flash memory.

Key reconstruction
When the device needs to know $S$ again, it performs the following steps. It reads $W'$ and $\sigma'$ from flash (possibly modified by the attacker!). It does a PUF measurement, obtaining $X'$. It computes $S' = \text{Rep}(W', X')$ and then $\hat{\sigma} = f(S'||W')$. It verifies if $\hat{\sigma} == \sigma'$. If false, then it knows that something is wrong.

This method is secure if $S$ has enough entropy and if you believe in the security of the hash function $f$ (which in general is unproven).

Method based on a Message Authentication Code (MAC)

If you do not trust hash functions, there is an alternative that gives information-theoretic security. It is based on a MAC $g$ with the so-called ‘Key Manipulation Security’ (KMS) property [6]. Let $s$ be the MAC key, and $m$ a message. The authentication code is $\mu = g(s, m)$. You can think of it as a ‘symmetric-crypto signature’. Roughly speaking, a KMS-MAC achieves two types of security: (i) For fixed $s$ (unknown to the attacker) and known $(m, \mu)$, the attacker cannot forge an acceptable pair $(m', \mu')$, in the sense that his probability of success is just the probability of guessing correctly. This is the ordinary property of an information-theoretically secure MAC.

(ii) Even if the attacker is able to induce changes $s \rightarrow s' = s + \Delta$, with known $\Delta$ (but still unknown $s$), he is still unable to make forgeries $(m', \mu')$ that are consistent with $s'$.

Initialization
The device does a PUF measurement, obtaining $X$. It computes $(S, W) = \text{Gen}(X)$. It splits $S$ into two independent parts, $S_1$ and $S_2$. The role of $S_1$ is authenticating the helper data, while $S_2$ is the memory encryption key. The device computes $\mu = g(S_1, W)$. It stores $W, \mu$ in flash.

Key reconstruction
When the device needs to know $S_2$ again, it performs the following steps. It reads $W'$ and $\mu'$ from flash (possibly modified by the attacker!). It does a PUF measurement, obtaining $X'$. It computes $S' = \text{Rep}(W', X')$ and splits $S'$ into $S'_1, S'_2$. It checks if $\mu' == g(S'_1, W')$. If false, then it knows that something is wrong.
5.4.7 Certified measurements

In these days of abundant processing power and great image manipulation techniques, how can you trust the integrity of any picture or video sequence? Or, in fact, data from any other kind of sensor?

We consider a sensor which is ‘irremovably’ embedded into a POK, i.e. in such a way that the key is damaged if one tries to separate the sensor from the POK. There is a certification authority whose public key $P_{CA}$ is known globally, and is embedded in the ROM of the device.

**Attack model**

An attack is considered successful if a manipulated sensor value passes verification. We expressly do not consider attacks where the ‘outside world’ is manipulated in order to obtain a specific sensor value; in that case the sensor is in fact working properly. (E.g. fooling a camera by placing a picture of an empty corridor in front of it.)

**Enrollment**

1. A random POK is created and mated to a sensor. The device has identifier $i$. It queries the POK and gets response $X$. It computes $(S, W) = \text{Gen}(X)$. The key $S$ is treated as a private asymmetric key. The device computes a public key $P$ from $S$. It outputs $i, W$ and $P$.

2. The CA has a descriptor $m$ for the device. (The descriptor may comprise sensor type, expiration date, etc.) The CA creates signatures $\sigma_W$ and $\sigma_P$ using its private key $S_{CA}$: $\sigma_W = \text{Sign}(S_{CA}, i||W)$ and $\sigma_P = \text{Sign}(S_{CA}, i||P||m)$. The CA gives $\sigma_W$, $\sigma_P$, $m$ to the device.

3. The device runs $\text{Verify}(P_{CA}, i||W, \sigma_W)$ and $\text{Verify}(P_{CA}, i||P||m, \sigma_P)$. If both signatures pass the test, the device stores $W, \sigma_W, P, m$ in its flash memory, which is considered insecure. Otherwise it generates an error message.

4. If there is no error message, the CA allows the sensor device to be used. The ‘certificate’ $Z_i = (i, P, m, \sigma_P)$ is published.

**Sensor device operation**

When the device is activated, it accesses its ROM to retrieve $i, P_{CA}$ and its flash memory to retrieve $W, P, m, \sigma_W, \sigma_P$. Then it runs $\text{Verify}(P_{CA}, i||W, \sigma_W)$ and $\text{Verify}(P_{CA}, i||P||m, \sigma_P)$. It only proceeds if both checks are passed. The device does a PUF measurement, obtaining $X'$. It computes $S' = \text{Rep}(X', W)$. From $S'$ it computes a public key $P'$. The device only proceeds if $P' = P$.

Sensor data $V$ from the device is always accompanied by metadata: $i$ and $\sigma_V = \text{Sign}(S, V)$.

**Verification**

Someone receives the sensor data $V$ including the metadata $i$ and $\sigma_V$. He fetches $Z_i = (i, P, m, \sigma_P)$. He checks if $m$ corresponds to the type of sensor data. He runs $\text{Verify}(P_{CA}, i||P||m, \sigma_P)$ and $\text{Verify}(P, V, \sigma_V)$. He trusts the sensor data only if all these test are passed.

5.4.8 Controlled PUFs

Instead of binding a sensor to a PUF, you can also imagine binding a processor, including RAM memory and cache. Such a system is called a ‘Controlled PUF’ (CPUF) in the literature. (We will call it a CPOK when the PUF has only one CRP.) It is a trusted computing environment; you can outsource code execution to it and be sure of the result’s integrity.

**Exercise 5.4** In analogy with the certified measurement scenario, work out the details of how a CPOK would operate: (a) Enrollment. (b) Verification of the Operating System (OS) code by the CPOK, assuming that the OS is stored in insecure memory. (c) Communication between a user and the CPOK, such that both the program code and the result are inaccessible to eavesdroppers. (d) Verification of the result’s integrity by the user.
5.4.9 Reconfigurable POKs for Trusted Computing

The concept of a CPOK can be carried one step further. The Trusted Computing Group (TCG) has recommendations for a so-called Trusted Computing Platform (TCP) for PCs. One of the requirements is that the platform should be able to securely store state information (in insecure memory) between powering down and powering up. Such state information is very important, as it comprises event counters, rights counters and other time-dependent data. The most dangerous attack to guard against is the replay attack: the state information is replaced by an old state in which some rights have not been used up, money not yet spent etc.

The best defense against replay attacks is to have at least one secure counter inside the TCP, which can be used to check some time stamp in the state information. Such a secure counter may be implemented in the form of a key which is regularly refreshed randomly. If standard digital storage of critical data is not trusted (that’s why the CPOK concept was introduced), then one is automatically led to the conclusion that a special kind of POK is needed: one that can be re-set to a new random (uncontrollable) key value. This concept may be called a ‘Reconfigurable PUF’.

Exercise 5.5 Sketch how the storage of state information would work if the TCP contains a reconfigurable POK (RPOK). Make sure that (a) replay attacks are thwarted, and (b) switching off the power during a state update does not lead to data loss. You may assume that the TCP also contains an ordinary POK.

5.4.10 Software-to-hardware binding

Lots of European and American companies outsource the manufacture of electronic devices to Asian countries. Unsurprisingly, this trend facilitates the appearance of cloned devices in the market. Apart from the hardware (HW) also the software (SW) is cloned; cloning SW is particularly easy as it just consists of copying digital information. We consider the case of firmware running on an FPGA\(^6\). A company can spend a great deal of effort (money) writing good firmware, and then see it getting copied in an instant. The solution is to ‘bind’ the firmware to the FPGA,

\(^6\)Field Programmable Gate Array, a form of programmable HW. Components of the FPGA can be configured to behave as a gate, memory cell, etc. When an FPGA is powered up, it loads a configuration from an externally stored file. Using an FPGA instead of a dedicated ASIC chip allows a company to achieve shorter product-to-market time and to update the firmware when necessary.
in the sense that the firmware is encrypted such that it can run only on one specific FPGA chip. (Not one kind or model of chip, but really one specific piece of silicon.) In order to achieve this, each FPGA needs to have a unique key. Say the FPGA with serial number $i$ has key $K_i$; then the stored configuration file intended for the $i$'th FPGA has to be encrypted with $K_i$.

How is $K_i$ stored in the FPGA? There are many options, mostly using standard digital means. However, read-proof storage is hard to achieve, so PUF/POK-like schemes could be a good option. For cost reasons the PUF should not take the form of additional hardware, but simply involve a measurement of the existing FPGA hardware. (Note: due to this restriction a real POK is excluded. The attacker has full access to this PUF.)

This gives a bit of a chicken-and-egg situation: The attacker knows that a key is derived from the FPGA hardware, which he himself also has access to. So how do you hide from him which part of the FPGA is probed and which algorithm is then used to compute $K_i$? The program/configuration that performs these tasks has to be delivered in unencrypted form! Here the firmware developers have a bit of unexpected luck. It turns out that the file format (“bitstream”) in which FPGA configurations are stored is highly obfuscated, even when there is no secrecy objective. Rather simple configurations already give rise to Bitstream files that are difficult to reverse-engineer. So due to this piece of luck the following scheme can be realized. (i) The FPGA is powered up. It loads a plaintext (but highly obfuscated) Bitstream file $N_1$ from insecure EEPROM and executes it. (ii) The $N_1$ program does a measurement of the FPGA hardware. It combines this with helper data to obtain a key $K_i$. (iii) The FPGA loads an encrypted Bitstream file, $N_2$, and the $N_1$ program decrypts it.

Variations to this scheme are of course possible, e.g. a single obfuscated program that cannot run properly except if some measurement of the FPGA yields exactly $K_i$. The security depends among others on the quality of the obfuscation.

**Exercise 5.6** Consider an FPGA whose SRAM is used to extract a ‘fingerprint’ for the purpose of software-to-hardware binding. An attacker does a ‘cold boot’ attack [16] on an FPGA.
(a) How much does he learn about the ‘fingerprint’?
(b) The attacker’s aim is to copy the FPGA configuration file to another FPGA and get it to work. How much does the above attack help him in achieving his aim?

### 5.4.11 Remote authentication of Public PUFs

Even if the properties of a PUF are publicly known, it can still be used for remote authentication under certain conditions.

Imagine the following scenario: There is a class of PUFs (e.g. a variant of silicon PUFs) which supports an exponentially large challenge space, but is emulatable. Suppose that the parameters of every PUF have been published, so that anybody can emulate PUFs, i.e. it is possible to compute the response to any challenge without having access to a PUF. Suppose further that the emulation takes significantly more time than the physical response time of the PUF. Such PUFs have been dubbed ‘Public PUFs’.

Under these circumstances, an authentication scheme can be built based on the ability of a PUF to produce responses quickly.

Note that the response from a Public PUF should not be used to generate a secret key! After a time delay, everybody has access to the emulation results. (And therefore a challenge should never be used more than once.) However, it does make sense to authenticate messages using a MAC key. During the time delay, the owner of the PUF is the only party capable of MAC-ing a message using that key.

In [20] an FPGA-based Public PUF was proposed. It makes use of the enormous number of possible ways in which an FPGA can be reconfigured, as well as the difficulty of reverse-engineering configuration files.
5.5 Entropy of PUFs

In Section 5.4.1 we talked about the entropy of a PUF in a very loose way. Here we treat this subject a bit more formally.

We start by defining a ‘PUF space’ $\mathcal{K}$ for a certain type of PUF. This is the configuration space for all the random properties in a PUF. $\mathcal{K}$ is a finite set in which every element corresponds to one possible PUF configuration. When a PUF is created randomly, we can represent this as a stochastic variable $K \sim \mathcal{K}$.

How can the set be finite if the random properties, e.g. particle locations, are continuous? The answer is finite resolution. There is no measurement that gives an infinite number of accurate digits after the comma; hence everything becomes discrete (but very fine-grained if the measurements are very accurate.)

In general, not every configuration of particles etc. occurs with equal probability. There is a probability distribution $P$ on $\mathcal{K}$, with

$$p_k = P(k) = \Pr[K = k].$$  \hfill (5.11)

From this we can define the creation entropy of a PUF as

$$\text{Creation entropy: } H(K) = - \sum_{k \in \mathcal{K}} p_k \log p_k.$$  \hfill (5.12)

Next we define a finite set $\mathcal{M}$ to be the set of all ‘allowed’ measurements that can be performed on the PUF.\(^7\) The notation $\mathcal{M}$ is rather abstract; the elements of $\mathcal{M}$ are detailed descriptions of experiments. We define $\mathcal{U}_m$ as the set of possible outcomes of a measurement $m \in \mathcal{M}$.

Given these definitions, we can now say that a measurement $m \in \mathcal{M}$ is in fact a mapping from $\mathcal{K}$ to $\mathcal{U}_m$. When $m$ is applied to a PUF $k \in \mathcal{K}$, the result is a measurement outcome $m(k) \in \mathcal{U}_m$.

$$m : \mathcal{K} \to \mathcal{U}_m \quad k \mapsto m(k).$$  \hfill (5.13)

Two measurements, say $m_1$ and $m_2$, can be combined into a single measurement $\mu \in \mathcal{M}$ simply by putting their outcomes together in a vector: $\mu(k) := (m_1(k), m_2(k))$. This is straightforwardly generalized to more than two measurements. Thus the set $\mathcal{M}$ accommodates not only simple experiments, but also very elaborate ones that require multiple steps.

Now consider a fixed $m$ being applied to a randomly created unknown PUF. We introduce the notation $I_m$ for the entropy of this experiment. (We assume for the moment that there is no measurement noise.) We call this the ‘noiseless measurement entropy’ of $m$. We have

$$\text{Noiseless measurement entropy: } I_m = H(m(K)).$$  \hfill (5.14)

---

\(^7\)E.g. various ways of probing an optical PUF with optical wavelengths, but not ultraviolet or even shorter wavelengths.
The probability distribution of the RV \( m(K) \) is computed as follows. For any \( m \in \mathcal{M} \) and \( u \in \mathcal{U}_m \)
\[
\pi_{m,u} = \Pr[m(K) = u] = \sum_{k: m(k)=u} p_k. \tag{5.15}
\]

The noiseless measurement entropy (5.14) can also be written as
\[
I_m = H(\{\pi_{m,u}\}_{u \in \mathcal{U}_m}) = -\sum_{u \in \mathcal{U}_m} \pi_{m,u} \log \pi_{m,u}. \tag{5.16}
\]

(Here the entropy \( H \) is expressed as a property of a distribution, while in (5.14) it is written as a property of a RV.)

Two measurements are called independent if \( I(m_1(K); m_2(K)) = 0 \), i.e. the outcomes depend on different ‘parts’ of \( K \).

Next we introduce measurement noise as an imperfect channel from \( u \in \mathcal{U}_m \) to \( u' \in \mathcal{U}_m \). Even though the ‘true’ outcome of the measurement is \( u = m(k) \), the noise in the equipment alters the result to some value \( u' \). For the probability of such an alteration we write
\[
t_{u \to u'} = \Pr[U' = u' | U = u]. \tag{5.17}
\]

This is the most general way of characterizing the noise. It allows the noise to depend on \( u \).

Noise may be caused by thermal fluctuations, quantization effects, stray electromagnetic fields, cosmic radiation, vibrations, voltage instabilities etc. Often noise is ‘additive’. This means that we can write \( U' = U + N \), where \( N \sim \rho_n \) is a RV such that \( t_{u \to u'} = \rho_n(u' - u) \), i.e. the probability of the noise amplitude is independent of \( u \). We have seen examples of additive noise in the previous chapters, e.g. Gaussian noise and the binary symmetric channel. Typically \( \mathbb{E}[N] = 0 \).

The noise level can be reduced by repeating a measurement several times. For the \( i \)'th repetition, \( i \in \{1, \cdots, \ell \} \), we have a noisy measurement result \( U'_i = m(K) + N_i \) where all the \( N_i \) are independent. Averaging these outcomes gives \( \frac{1}{\ell} \sum_i U'_i = m(K) + \bar{N} \), with \( \bar{N} := (1/\ell) \sum_i N_i \). It turns out that the amplitude of the averaged noise \( \bar{N} \) scales as \( 1/\sqrt{\ell} \). (You are asked to prove this in the next exercise.)

**Exercise 5.7** Consider additive noise with zero expectation value. Show that \( \mathbb{E}[\bar{N}^2] \) is proportional to \( 1/\ell \).

How many bits of information does one obtain about a PUF by doing one noisy measurement? The answer is given in the following lemma.

**Lemma 5.3 (PUF information revealed by a noisy measurement)** Consider an unknown PUF \( K \), a measurement \( m \in \mathcal{M} \) (\( m(K) = U \)) and noise that is not necessarily additive. The amount of information revealed by one noisy measurement outcome \( U' \) is
\[
I(U'; K) = I(U'; U).
\]

**Proof:** The left expression is exactly the information overlap we are looking for. We write
\[
I(U'; K) = H(U') - H(U'|K) = H(U') - H(U'|U) = I(U'; U). \]

**Corollary 5.4 (PUF information revealed by a noiseless measurement)** If there is no measurement noise then the revealed information is \( I_m \).

**Definition 5.5 (Measurable entropy of a PUF)** Consider a PUF space \( K \) and a space \( \mathcal{M} \) of allowed measurements. The **measurable entropy** of a PUF \( K \in \mathcal{K}, K \sim \mathcal{P} \), denoted by \( I^\text{meas}_{\mathcal{P}\mathcal{M}} \), is defined as
\[
I^\text{meas}_{\mathcal{P}\mathcal{M}} = \max_{m \in \mathcal{M}} H(m(K)).
\]
In words: the measurable entropy of a PUF is the amount of information revealed by the most accurate experiment available in $\mathcal{M}$. Note that this can be quite different from the creation entropy $H(K)$. In some cases there is much more entropy in a PUF than what gets revealed by the measurements that are done during standard operation. For instance, the creation entropy of a coating PUF is largely determined by the exact positions and orientations of TiN particles. The configuration space is huge. However, the sensor wires are only sensitive to the local capacitance, a single scalar that is an average effect of many degrees of freedom in the coating.

For bare PUFs it makes sense to define a ‘security parameter’ which measures how much effort it costs an attacker to characterize a PUF.

**Definition 5.6** Let $\mathcal{M}_0 \subseteq \mathcal{M}$ be the set of allowed measurements that can be performed in a single time step. The **security parameter** $S_{\mathcal{M}_0}$ of a bare PUF is defined as the minimum number of (non-compound) challenge-response measurements required to reveal all the measurable information of the PUF.

Def. 5.6 counts complicated measurements in $\mathcal{M}$ as multiple steps. The essence is that the security parameter is proportional to the time spent on the attack. After the attack, enough information about the PUF is available to have a substantial probability of giving a correct response to a fresh random challenge (without access to the PUF).

**Example 5.7** Consider an optical PUF consisting of two materials that are mixed perfectly at random. The challenges in $\mathcal{M}$ consist of probing the PUF with laser light of wavelength $\lambda$ in any way. The typical ‘non-compound’ measurement of Def. 5.6 is taking a photo of the speckle pattern after transmission. The illuminated area on the PUF is $W^2$. We set the thickness $d$ to be much larger than the mean scattering length $\ell$. Since light is not able to ‘see’ details far below $\lambda$, it makes sense to divide the PUF into small cubes (‘voxels’) of volume $\lambda^3$ and indicate by a 0/1 which material is predominant in that cube. The number of voxels is $N_{\text{vox}} = W^2d/\lambda^3$. The measurable entropy$^8$ cannot exceed $N_{\text{vox}}$.

Naively one would expect the entropy of a speckle pattern to be of order $N_{\text{mod}} \log(\#\text{grayscales})$, where $\#\text{grayscales}$ is the number of light levels that can be distinguished by ‘perfect’ equipment, given the luminosity of the laser. However, it turns out that the multiple scattering causes dependence between the modes: the information contained in the speckle pattern is less than the sum of the entropies per mode. Computing the exact entropy is nontrivial. In [33] it was found that $I_m \propto (\ell/d) \cdot N_{\text{mod}} \log(\#\text{grayscales})$.

Consider challenges in the form of a parallel laser beam under a certain angle, where the angle is the actual challenge. The number of different challenges that lead to a different response is $N_{\text{mod}}$. Now the product $N_{\text{mod}} I_m$ is larger than $N_{\text{vox}}$. (Why?) Hence, if you are somehow able to compose challenges that reveal independent information about the PUF (best case assumption for the attacker), then you need far fewer than $N_{\text{mod}}$ challenges to implement a successful attack. We can conclude that

$$S_{\mathcal{M}_0} \propto \frac{N_{\text{vox}}}{I_m} \propto \frac{d^2}{\lambda\ell \log(\#\text{grayscales})}.$$  \hspace{1cm} (5.18)

This is a pessimistic result, as it is based on an assumption that is very much in favour of the attacker.

**Exercise 5.8** In the above example, the assumption of perfect mixing is not very realistic. (The content of a voxel is not really independent of neighbouring voxels.) Replace it by a more realistic assumption and argue (in words) how it affects the analysis of the security parameter.

**Example 5.8** Consider one sensor in a coating PUF. Assume that it can measure capacitance only at one frequency. There is only one challenge. No matter how large or small the measurable entropy is, the response reveals this whole entropy.

---

$^8$Note that we do not actually compute the measurable entropy according to its definition; instead, we upper bound it using general knowledge about optical probing.
Chapter 6

Fuzzy Extractors and Secure Sketches

You have already seen a very informal introduction to Fuzzy Extractors (FEs) in Section 5.2. In this chapter we define FEs and Secure Sketches (SSs) more formally and then show examples for various types of measurement data. FEs and SSs are also known as ‘helper data schemes’.

6.1 Definitions

Definition 6.1 A Fuzzy Extractor for a source space $\mathcal{X}$ and target space $\{0,1\}^*$ consists of a (possibly nondeterministic) procedure $Gen: \mathcal{X} \rightarrow \{0,1\}^* \times \{0,1\}^*$ : $x \mapsto (s_x, w_x)$ (‘generate’) which extracts a secret $s_x$ and helper data $w_x$ from $x$, and a procedure $Rep: \mathcal{X} \times \{0,1\}^* \rightarrow \{0,1\}^*$ : $(x', w) \mapsto s_{x'}$ (‘reproduce’) which tries to reproduce the secret from the helper data and a fresh measurement $x'$. A fuzzy extractor must satisfy the following properties:

- **Correctness:** The probability that $s'_{x} = s_{x}$ must be close to 1.
- **Security:** The random variable $S_x$ must be close to uniform, given knowledge of $W_x$.

In the literature there are various options for the exact definition of Correctness. Some use the tolerable Hamming distance $d(x, x')$ as a criterion. Others specify the probability of correct reconstruction.

Similarly, there are various ways to define the Security property. Some use the conditional Shannon entropy $H(S_X|W_X)$, some use conditional min-entropy.

Definition 6.2 A Secure Sketch for a discrete source space $\mathcal{X}$ consists of two algorithms, SS: $\mathcal{X} \rightarrow \{0,1\}^* : x \mapsto w_x$ (“sketch”) and Rec: $\mathcal{X} \times \{0,1\}^* \rightarrow \mathcal{X} : (x', w_{x}) \mapsto \hat{x}$ (“reconstruct”), with correctness and security defined as follows. A Secure Sketch must satisfy the following properties:

- **Correctness:** The probability that $\hat{X} = X$ must be close to 1.
- **Security:** $X$ given $W_X$ must have high entropy.
Again, the exact definition of Correctness and Security can be specified in several ways. We will not elaborate on this. If $X$ is a discrete RV, then there exists a generic FE construction from a Secure Sketch and a strong extractor. The Secure Sketch makes it possible to exactly reconstruct $X$ from $X'$ and the helper data; after that the strong extractor extracts a key from $X$.

**Exercise 6.1** Explain what a secure sketch for a continuous random variable would look like. (You may assume that $X$ can be measured without noise, but not $X'$.) Would it be practical?

**Exercise 6.2** Make a schematic drawing how the Gen and Rep algorithms of a fuzzy extractor can be realized using the SS and Rec algorithms of a Secure Sketch. Explicitly draw how the Gen and Rep components contain SS and Rec.

**Definition 6.3 (Robust fuzzy extractor)** We slightly broaden the definition of a fuzzy extractor. We use the notation $\pi$ for the ‘public data’; this includes redundancy data as well as additional data such as hashes, MACs, signatures etc. The output of the Rep algorithm is now in $\{0, 1\}^\ell \cup \bot$, where the ‘bot’ symbol $\bot$ denotes an error message. A fuzzy extractor is called $(n, \delta)$-robust if an adversary has success probability $\epsilon$ in the following game:

- $X$ is measured, and $(s, \pi) = \text{Gen}(x)$ is computed. The adversary receives the public data $\pi$.
- The adversary outputs $n$ strings $\pi_1, \ldots, \pi_n \neq \pi$.
- The adversary wins if $\exists i: \text{Rep}(x_i, \pi_i) \neq \bot$, where the $x_i$ are repeated noisy measurements of $X$.

In words: The Rep algorithm may now also abort instead of outputting an $\ell$-bit key. The adversary tries to manipulate the public data (including the authenticating signature/hash/MAC) in such a way that Rep is tricked into computing a key using the fake public data and thinking it is the correct key.

In Sections 5.4.5 and 5.4.6 we have already seen several examples of Robust fuzzy extractors (without naming them as such). The scheme in Section 5.4.5 has a Certification Authority (CA) which signs the helper data and the PUF public key using its own private key $S_{CA}$. Here $\pi = (w, \text{Sign}(S_{CA}, w), \text{Sign}(S_{CA}, \text{PUF publickey}))$. The $\text{Rep}$ procedure verifies the signatures and outputs $\bot$ if they are incorrect.

The first scheme in Section 5.4.6 relies on an ‘ideal’ hash function $f$ (i.e. the “random oracle model”) to authenticate the redundancy data while hiding the PUF key. Here $\pi = (w, f(s||w))$.

The second scheme in Section 5.4.6 is information-theoretically secure (needing no assumptions). It makes use of a recently developed kind of message authentication code with the Key Manipulation Security property [6]. Without giving the exact definition of the KMS property, we mention that ordinary MACs cannot be proven secure when the key is under attack, instead of only the message and the authentication code. In the setting of fuzzy extractors, the attacker manipulates the redundancy data, which then gets used to extract a key which is used to MAC the redundancy data ... rather circular. A KMS-MAC is provably secure under such an attack. The key $s$ is split into parts $s_1, s_2$. The KMS-MAC $\mu$ uses $s_1$ as its key, $\pi = (w, \mu(s_1, w))$. The leftover $s_2$ is the actual secret key.
6.2 When to use the FE and SS primitives

The main difference between a FE and a SS lies in what they try to achieve. The aim of the FE is to reliably extract a cryptographic key from noisy data. The aim of the SS is to reliably extract a string with sufficient (min-)entropy. Sometimes the second objective is sufficient, e.g. for the hashing of noisy passwords or biometric data.

Since the requirements of a SS are less stringent, it is easier to construct a SS than a FE; in general a SS extracts more (min-)entropy than a FE from the same source.

<table>
<thead>
<tr>
<th>Application</th>
<th>privacy of X?</th>
<th>uniform secret?</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>password authent.</td>
<td>✓</td>
<td></td>
<td>One-Way Function</td>
</tr>
<tr>
<td>biometric authent.</td>
<td>✓</td>
<td></td>
<td>Secure Sketch + OWF</td>
</tr>
<tr>
<td>anticounterfeiting PUF</td>
<td>✓</td>
<td></td>
<td>Secure Sketch + OWF</td>
</tr>
<tr>
<td>anticounterfeiting PUF</td>
<td>●</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>PUF authent. w/o MACs</td>
<td>●</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>PUF authent. with MACs</td>
<td>●</td>
<td></td>
<td>Fuzzy Extractor</td>
</tr>
<tr>
<td>POK</td>
<td>●</td>
<td></td>
<td>Fuzzy Extractor</td>
</tr>
</tbody>
</table>

Figure 6.1: Overview of applications and the best primitive to use (Fuzzy Extractor, Secure Sketch, or neither). A tick mark ✓ indicates if privacy/uniformity is considered important.

Exercise 6.3 Consider a system in which first all PUFs are enrolled; authentication of PUFs is done only after all the enrollments are finished. In other words, once the system is in use, no more PUFs are enrolled.

(a) Is it possible to have a secure error correction method that does not need individual helper data? [It is not allowed to discard any PUFs.]

(b) Same question, but now new enrollments can happen at any time.

6.3 The Code Offset Method

6.3.1 The original Code Offset Method

Let \( C \) be a linear binary error correcting code with message space \( \{0,1\}^k \) and codewords in \( \{0,1\}^n \). It has an encoding algorithm \( \text{Enc}: \{0,1\}^k \rightarrow \{0,1\}^n \) and a decoding algorithm \( \text{Dec}: \{0,1\}^n \rightarrow \{0,1\}^k \). Let the PUF/biometric measurement space be \( \mathcal{X} = \{0,1\}^n \). The so-called Code Offset Method (COM) works as follows.
COM Enrollment phase: The Gen algorithm takes the enrollment measurement $x$ as input. It generates a random string $s \in \{0,1\}^k$ and encodes it to $c_s = \text{Enc}(s)$. It outputs $s$ as a secret and

$$w = c_s \oplus x$$

as helper data.

COM Reconstruction phase: The Rep algorithm takes a fresh measurement $x'$ and $w$ as input. It computes and outputs

$$\hat{s} = \text{Dec}(x' \oplus w).$$

![Figure 6.2: Artist impression of the code offset method. The plane of the drawing represents $\{0,1\}^n$. The grid points represent the codewords. The helper data $w$ is an arrow pointing from $x$ to the codeword $c_s$. Noise $x' - x$ around $x$ effectively becomes noise centered on $c_s$, shown as the grey disc around $c_s$. The Dec algorithm corrects the noise provided it’s not too strong.]

An intuitive picture is shown in Fig. 6.2. The helper data can be seen as a shift (offset) that maps $x$ onto a random codeword. (In general $x$ itself is not a codeword.) Any noise on $x$ translates to noise around $c_s$, which is easy to correct.

**Exercise 6.4** Can you think of a reason why the Code Offset Method picks an arbitrary codeword $c_s$ as the secret instead of simply taking the codeword nearest to $x$? [Note: the answer to this question cannot be found in these lecture notes.]

**Exercise 6.5** Consider the Code Offset Method with $X$ uniformly distributed. Show that $H(S|W) = H(S)$. Can the Code Offset Method be used as a Fuzzy Extractor in this case? Can it be used as a Secure Sketch?

**Exercise 6.6** Consider again the COM, but now $X$ is not uniformly distributed. Derive an expression for $H(S|W)$. Is it still true that $H(S|W) = H(S)$? Can the Code Offset Method be used as a Fuzzy Extractor in this case? Can it be used as a Secure Sketch?

### 6.3.2 The Syndrome-Only Code Offset Method

There is a variant of the COM that does not need the random codeword. It uses an error-correcting code with syndrome $\text{Syn} : \{0,1\}^n \rightarrow \{0,1\}^{n-k}$ The code allows for syndrome decoding $\text{SynDec} : \{0,1\}^{n-k} \rightarrow \{0,1\}^n$ (see Section 2.6.3). The syndrome decoding finds an $n$-bit error pattern matching the $(n-k)$-bit syndrome. The scheme is called the Syndrome-Only COM, and serves as a Secure Sketch.

**Syndrome-Only COM Enrollment phase:** The SS algorithm takes the enrollment measurement $x$ as input. It computes

$$w = \text{Syn} x$$

and outputs $w$ as helper data.
6.4. FUZZY EXTRACTOR BASED ON PARTITIONS

**Syndrome-Only COM Reconstruction phase:** The Rec algorithm takes a fresh measurement $x'$ and $w$ as input. It computes

$$\hat{x} = x' \oplus \text{SynDec}(w \oplus \text{Syn } x').$$

(6.1)

and outputs $\hat{x}$.

The reason why (6.1) yields $\hat{x} = x$ (for reasonable amount of noise) is that the syndrome function $\text{Syn}$ is linear. We have $w \oplus \text{Syn } x' = \text{Syn } x \oplus \text{Syn } x' = \text{Syn } (x \oplus x')$, i.e. the syndrome of the error pattern $x \oplus x'$. Running $\text{SynDec}$ returns the error pattern itself, and xor-ing this into $x'$ yields $x$.

**Exercise 6.7** Show that the following holds for the Syndrome-Only COM,

$$I(X;W) = H(\text{Syn } X) \quad H(W|X) = 0 \quad H(X|W) = H(X) - H(\text{Syn } X).$$

(6.2)

**Exercise 6.8** Consider a weak device that has to run the Rec algorithm of the Syndrome-Only COM. The $\text{SynDec}$ computation is the most difficult operation.

Is it feasible, from a security point of view, to outsource the $\text{SynDec}$ computation? In other words, is it dangerous for the device to reveal $w \oplus \text{Syn } x'$?

**Remark:** Any Secure Sketch can be used as the basis for a Fuzzy Extractor (see exercise 6.2). Hence, even for non-uniform $X$ we can build a FE with the (Syndrome-only) COM as the main building block.

### 6.4 Fuzzy extractor based on partitions

We briefly describe the ‘partitioning’ method [35] of constructing a FE for a continuous RV. The basic idea is to sacrifice the least significant bits of $X$ for use as helper data, so that the most significant bits are noise-resilient and secret. Let $X \sim \mathbb{P}$. The space $\mathcal{X}$ is split up (‘partitioned’) into $n$ equiprobable regions $\{\mathcal{A}_s\}_{s=0}^{n-1}$, i.e. $\Pr[X \in \mathcal{A}_s] = \mathbb{P}(\mathcal{A}_s) = 1/n$. The numerical labels of these partitions correspond to the secret $s$. Within each partition, there is a further partitioning into $m$ equiprobable regions $\mathcal{A}_{sw}$; these are numbered $w \in \{0, \cdots, m-1\}$, and we have $\Pr[X \in \mathcal{A}_{sw}] = \mathbb{P}(\mathcal{A}_{sw}) = 1/(nm)$. The subpartition label is the helper data.

**Enrollment phase:** $X$ is measured, and it is determined in which interval $\mathcal{A}_{ij}$ the measured value $x$ is situated. The secret is set to $s_x = i$, and the helper data to $w_x = j$.

**Reconstruction phase:** $X'$ is measured. It is determined for which $s'$ the interval $\mathcal{A}_{s',w_x}$ is closest to $x'$. This $s'$ is the reconstructed key.

For an attacker who knows $w$, the secret $S$ is uniformly distributed,

$$\Pr[S = s|W = w] = 1/n.$$  

(6.3)

Hence $H_\infty(S|W = w) = \log n$ and $H_\infty(S|W) = \log n$.

In practice, the choice of $n$ and $m$ depends on the noise characteristics. If $n$ is chosen too large, then the error correction fails. The parameter $m$ can, in principle, be chosen infinitely large. The error correction improves with increasing $m$. However, the improvement per step $m \rightarrow m + 1$ rapidly dwindles down.

**Exercise 6.9** Can the partitioning scheme be applied to discrete random variables?

**Exercise 6.10** Explain why increasing $m$ improves the error correction.
Figure 6.3: Example of the partitioning scheme with $X \in \mathbb{R}$ normal-distributed, $w \in \{0, 1, 2\}$ and $s \in \{0, 1\}^2$.

Figure 6.4: Example of the partitioning scheme with $X \in \mathbb{R}^2$, $w \in \{0, 1\}^2$ and $s \in \{0, \cdots, 8\}$. The subpartition $w = 1$ is shown in green.
6.5 A fuzzy extractor based on universal hash functions

We have seen Universal Hash Functions (UHFs) in Section 3.4. UHFs provide a compression that makes a RV more uniform. UHFs do not have the same security properties as cryptographic hash functions.

We present a fuzzy extractor scheme in which all three functionalities are realized using UHFs: information reconciliation (error correction), privacy amplification (making the secret more uniform) and robustness (protecting the integrity of the public data). It is assumed that there is a source of public randomness which is available for lookup (Common Reference String). The noise is assumed to have special characteristics: given a noisy \( x_0 \), the number of candidate \( x \) values is limited, so that an exhaustive search for \( x \) is feasible. We introduce the notation \( \mathcal{B}_{1-\theta}(x') \) for the set of all neighborhoods of \( x' \) such that \( x \) is contained in the neighborhood with probability at least \( 1 - \theta \).

![Diagram](Figure 6.5: The Gen stage of the fuzzy extractor. Three different (almost) universal hash functions are applied to the input \( X \).)

**System setup phase:**

Alice and Bob beforehand agree on three\(^1\) almost universal families of hash functions \( \{\Phi_r\}_{r \in R} : \mathcal{X} \to \{0,1\}^c \), \( \{\Psi_t\}_{t \in T} : \mathcal{X} \to \{0,1\}^k \) and \( \{\Gamma_j\}_{j \in J} : \mathcal{X} \to \{0,1\}^\sigma \). (See Fig. 6.5.) These are \( 2^{-c}(1 + \delta_\Phi) \), \( 2^{-k}(1 + \delta_\Psi) \) and \( 2^{-\sigma}(1 + \delta_\Gamma) \) almost universal, respectively. The Common Reference String (CRS) consists of the random values \( \{r, t, j\} \). Alice and Bob also agree on a function \( F : \{0,1\}^\sigma \times \{0,1\}^* \to \{0,1\}^m \) that uses a \( \sigma \)-bit key to produce an \( m \)-bit authentication code. The \( \Phi, \Psi, \Gamma \) hash families are known to the attacker, as are \( c, \sigma, k, F \) and the CRS.

**Enrolment phase:**

1. Alice performs a measurement and obtains an outcome \( x \).
2. She computes \( s = \Phi_r(x) \), \( w = \Psi_t(x) \), \( v = \Gamma_j(x) \) and \( a = F(v, w) \).
3. She stores \( w \) and \( a \) in nonvolatile memory.

**Attack phase:**

The attacker modifies \( \{w, a\} \) to \( \{\tilde{w}, \tilde{a}\} \).

**Reconstruction phase:**

1. Bob reads \( \{\tilde{w}, \tilde{a}\} \) from the nonvolatile memory and \( \{r, t, j\} \) from the CRS.
2. Bob performs a measurement and obtains an outcome \( x' \). He chooses a neighborhood \( B \in \mathcal{B}_{1-\theta}(x') \). He compiles a list \( L = \{x_i \in B : \Psi_t(x_i) = \tilde{w}\} \). If \( L = \emptyset \), the protocol aborts in failure.
3. For all \( x_i \in L \), Bob computes \( v_i := \Gamma_j(x_i) \). He checks if \( F(v_i, \tilde{w}) = \tilde{a} \). In the event that a single match \( x^* \) occurs, the protocol is considered to have succeeded, and Bob accepts \( \hat{s} := \Phi_r(x^*) \) as the reconstructed shared secret. If there are no matches, or more than one, then the protocol aborts in failure.

\(^1\)One may also take a single AUHF and split the output into three parts. However, the construction with different functions allows us to choose their properties independently.
CHAPTER 6. FUZZY EXTRACTORS AND SECURE SKETCHES

Theorem 6.4 Let $\delta$ be defined as $1 + \delta = (1 + \delta_\Phi)(1 + \delta_\Psi)(1 + \delta_\Gamma)$. In the above given Fuzzy Extractor scheme, if the parameters satisfy

$$c + k + \sigma \leq H_2(X) + 2 - \log \frac{1}{\varepsilon^2 - \delta/4}$$

(6.4)

then

$$\Delta(RTJWA S; RTJWA U_c) \leq \varepsilon.$$  (6.5)

Proof: $A$ is a function of $R$, $T$, $J$, $W$, $V$, hence the combined variable $RTJWA$ is a function of the combined variable $RTJWV$. We use the fact that applying a function cannot increase the statistical distance. Thus

$$\Delta(RTJWAS; RTJWAU_c) \leq \Delta(RTJWVS; RTJWVVU_c).$$  (6.6)

Next, for any random variables $X \in \mathcal{X}, Y \in \mathcal{Y}$ it holds that $\Delta(XY; U_{X,Y}) \leq \Delta(XY; U_{X,X'})$, where $U_X$ is a variable uniform on $\mathcal{X}$. This gives

$$\Delta(RTJWVS; RTJWVVU_c) \leq \Delta(RTJWVS; RTJU_{k+\sigma+c}).$$  (6.7)

Now consider the concatenation $F_{rtj}(x) := \Phi(x)||\Psi(x)||\Gamma(x)$. This behaves as an AUHF with ‘almost’-parameter $\delta$. [You are asked to show this in Exercise 6.11.] We want to upper bound the last expression in (6.7) by $\varepsilon$. Using the leftover hash lemma (Theorem 3.12) completes the proof.

\[\Box\]

Exercise 6.11 Prove the concatenation property of AUHFs that was used in the proof of Theorem 6.4. In other words, prove that $F_{rtj}(x)$ is an almost-universal hash function with $\eta_F = 2^{-(c+k+\sigma)}(1 + \delta)$.

6.6 A helper data scheme for coating PUFs

As we have seen in Section 5.3.2, the response of a coating PUF consists of a number of capacitance values. These capacitances are continuous RVs, and their signal-to-noise ratio (see Eq. 2.50) is such that typically 3 to 5 bits can be extracted reliably per capacitor.

The most straightforward way to construct a FE for a coating PUF is to apply the partitioning method of Section 6.4, and to set the $n, m$ parameters such that the error probability of $Rep$ is very low for each capacitor. Let’s say there are $N$ capacitors labeled $0, \cdots, N - 1$ and the keys derived from them are $s_0, \cdots, s_{N-1} \in \{0, \cdots, n - 1\}$. The total key of all the capacitors together can then be constructed as an integer $S_{tot} = \sum_{i=0}^{N-1} S_in^i \in \{0, \ldots, n^N - 1\}$. The min-entropy is $H_\infty(S_{tot}) = N \log n$.

The straightforward way may not be the most efficient; it may be possible to extract more bits, depending on the exact noise properties. In the above scheme, $n$ has to be rather small because otherwise the $s_i$ are too noisy. Let us choose a larger value, $n = 2^b$, where $b$ is some integer, and again apply the partitioning method. Now each capacitor yields exactly $b$ extracted bits, but they are too noisy to be accepted as the final key. Hence one additional error correction step is needed.
6.7. A HELPER DATA SCHEME FOR OPTICAL PUFs

1. The labeling of the $s_i$ values is done according to a Gray code (Fig. 6.6). This ensures that an error in $s_i$ caused by a shift over one interval leads to only one bit flip. (And only two bit flips if the shift is over two intervals.)

2. All the $N$ Gray codewords are concatenated together into a bitstring of length $N_b$.

3. We apply the code offset method, using a code with message space $\{0,1\}^k$ and codewords in $\{0,1\}^{N_b}$.

It can happen that the $k$ in this elaborate scheme is larger than $H_\infty(S_{tot})$ in the simple scheme, for identical error rate in the final key.

![Figure 6.6: Equiprobable partitions with Gray code labeling.](image)

**Exercise 6.12** Is the key in the combined scheme (partitioning method + Gray code + code offset method) uniform, given knowledge of the public data?

6.7 A helper data scheme for optical PUFs

One can think up many different helper data schemes for optical PUFs. The main challenge is to cope effectively with the error patterns specific to speckle. Without providing an explanation we mention the most prominent properties of noise in optical PUFs. First, small misalignments of the laser beam w.r.t. the PUF cause small shifts and rotations of the image. Furthermore, somewhat larger misalignments cause random-looking brightness changes in the speckles. Finally, misalignments of the order of the angle between modes, or larger, cause the speckle pattern to change beyond recognition. Ideally, the helper data allows Rep to decide if the speckle pattern $X'$ is sufficiently close to $X$ even before key reconstruction begins. (Unnecessary work can be avoided in that way.)

We sketch two examples of how this can be achieved.

1. One part ($w_1$) of the speckle pattern image $x$ is ‘sacrificed’: it is made public and serves as helper data. A part of $x$ close to $w$ is completely discarded. (It carries information strongly correlated to $w$.) What is left over is secret. From this part a bitstring is derived using some image processing algorithm. A popular method in the literature [23] is to apply a Gabor filter, roughly speaking a direction-sensitive edge detection algorithm. (This filter is also applied in iris recognition [7].) The largest Gabor coefficients are selected as ‘reliable components’, and only their signs are kept as binary secrets; the rest of the information is discarded. Pointers to the reliable components are stored as part of the helper data ($w_2$). Finally the code offset method is applied to the bitstring in order to get rid of residual errors. This step also adds to the helper data ($w_3$).

2. The first part of the helper data ($w_1$) consists of an image or list that contains the most prominent features of $x$: the location and shape of the brightest/darkest/largest regions, but
NOT their brightness. The set of the brightnesses of these regions, each represented as a 0 or 1, forms a secret bitstring. Finally (if necessary) the code offset method is applied to this bitstring. The code offset method adds to the helper data ($w_2$).

**Exercise 6.13** The description above gives all the necessary ingredients for making a FE, but does not give a full prescription of all the steps. For method 1, write out in detail all the steps in Gen and Rep.

### 6.8 A helper data scheme for SRAM PUFs

The enrollment procedure Gen finds out which memory cells behave in a sufficiently reproducible manner. The first part of the helper data consists of pointers to such cells. Since the noise level is quite high, this is done in a special way. The pointers are clustered in groups of size $N$. All the pointers in a cluster point to cells with the same startup value. The Rep step does a ‘majority voting’ on each clusters, obtaining one bit per cluster. On top of all that, the code offset method is applied.

**Gen**
- Find the set $R$ of addresses of reliable cells. Initialize a vector $\vec{w}$ to $\emptyset$.
- For $i = 1...n$: {Generate a random bit $x_i$; Find $N$ different cells in $R$ whose content is $x_i$; append those cell addresses to $\vec{w}$ and remove them from $R$.}
- Generate random $s \in \{0, 1\}^k$; $\omega = \text{Encode}(s) - x$.
- Output $s$ as a secret and ($\vec{w}, \omega$) as helper data.

**Rep**
- For $i = 1...n$: {Read the next $N$ addresses from $\vec{w}$; Do a majority voting on their content in the SRAM, and call the result $x'_i$.}
- Append all the $x'_i$ to form a bitstring $x'$; Compute $s' = \text{Decode}(x' + \omega)$.

Making a good choice of ECC and code parameters ($n$, $N$) is nontrivial; it strongly depends on the required size of the secret ($k$) and on the amount of noise in the SRAM startup values.
Chapter 7

Distance bounding

7.1 Why distance bounding?

Sometimes an ordinary authentication protocol is not sufficient. Consider for instance the entrance gate to a high-security installation, with an access control system using wireless authentication devices. For the access control system it is not enough to know if it is talking to someone who knows the proper secret keys, it also has to know that this person is the one standing right in front of the entrance.

Authentication protocols are vulnerable to relay attacks. In one such attack (sometimes called the mafia fraud), the challenge is relayed to a different location, where a legitimate device is tricked into giving a response to the challenge; the response is relayed back to the verifier. (If the protocol requires mutual authentication, all messages for authentication in the other direction are relayed too.)

A variant of this attack is called the “terrorist fraud”. Now the legitimate device (again far away from the verifier) actually cooperates willingly with the attacker. However, the authentication secrets are not shared with the attacker; only derived quantities such as responses are shared.

There is an amusing story in [1] about the “Mig-in-the-middle attack”. Lore has it that in the late 1980s the Cuban air force, supporting the Angolans in their fight against Namibia, played the following trick on the South African Air Force (SAAF, supporting Namibia),

Several MIGs had loitered in Southern Angola, just north of the South African air defense belt, until of flight of SAAF Impala bombers raided a target in Angola. Then the MIGs turned sharply and flew openly through the SAAF’s air defenses, which sent IFF challenges. The MIGs relayed them to the Angolan air defense batteries, which transmitted them at a SAAF bomber; the responses were relayed back in real time to the MIGs, who retransmitted them and were allowed through (…)

---

1 In the case of simple RFID tags and smartcards this is trivial: the device will respond to any reader.
7.2 General design principles

The most trivial way to check the proximity of a device to the verifier is to demand galvanic contact or visual inspection. However, this completely negates all the advantages of a wireless system (no line of sight required, no insertion of cards into readers) and is often unacceptable.

Signal strength is a very unreliable measure of distance; the transmitter of a malicious device may be far stronger than that of legitimate devices. More importantly, even a correct distance derived from signal strength does not give any security against the mafia fraud and terrorist fraud attacks. The simplest way around all these problems is to measure signal round-trip time. The speed of light is finite; in flat empty space it is defined exactly as the constant
\[ c = 2.99792458 \cdot 10^8 \text{ m/s} \]
(\(\approx 1.1 \cdot 10^9\text{km/h}\)). Light travels some 300m every microsecond. The speed of light is a fundamental bound on the speed at which information can propagate.

**Hard fact:** If a device repeatedly proves itself able to correctly respond to an unpredictable challenge within time \( t \), counted from the moment when the verifier sends the challenge, then the location where the response is computed cannot be further away than \( x = c \Delta t / 2 \) from the verifier.

The word *unpredictable* is very important here. If the correct response is predictable, then some far away spoofer may start transmitting that response even before the challenge is sent.

Distance bounding in itself is not enough of course. It has to be combined with authentication. The challenged device has to prove that it knows a certain secret and that it can return the response within the required time. Let \( t_{\text{comp}} \) be the amount of time that it takes the device to compute the response from the challenge and the secret. Let \( x_{\text{max}} \) be the maximum tolerable distance. The maximum time within which the response has to arrive is

\[ t_{\text{max}} = 2 \frac{x_{\text{max}}}{c} + t_{\text{slack}}. \]  

(7.1)

Here \( t_{\text{slack}} \) is the amount of time that the verifier allows for the computation, plus a small tolerance margin to accommodate ‘noise’ such as random transmission delays, power fluctuations etc. If a device is sitting at the maximum range, then it has to finish its computation within time \( t_{\text{slack}} \), i.e. \( t_{\text{comp}} \leq t_{\text{slack}} \). An attacker with an infinitely fast processor is able to pretend to be within range \( x_{\text{max}} \) while instead his distance is \( x_{\text{spoofable}} \),

\[ x_{\text{spoofable}} = \frac{1}{2} c t_{\text{max}} = x_{\text{max}} + \frac{1}{2} c t_{\text{slack}}. \]  

(7.2)

It is immediately obvious from (7.2) that \( t_{\text{slack}} \) has to be extremely small. Hence, the computations occurring during \( t_{\text{slack}} \) cannot involve crypto operations such as signatures, block ciphers and hashes. Even worse, there is no time for the usual error correction algorithms that make the wireless communication noise-free. One is forced to use raw, noisy signals.

Then how is it possible to do a proper authentication? The answer is as follows: The protocol is split into a *slow phase* and a *quick phase*. In the slow phase, the device performs crypto operations on some initial challenge and prepares the responses to all possible challenges in memory.\(^2\) Then, in the quick phase, it receives very simple random challenges which only ask it to select a certain part of that memory and send that as a response. The verifier records the arrival time of the responses. After the quick phase, the verifier checks whether the responses arrived quickly enough and whether a sufficient number of responses is correct (allowing for the expected noise level).

Why is such a protocol secure against relay attacks? For the mafia fraud this is easy to answer. In the slow phase the responses stay in the legitimate device, which is far away; the attackers are forced to relay the quick-phase challenges to the legitimate device, which leads to a detectable time lag.

For the terrorist fraud the reasoning is more subtle, because the colluding legitimate device may leak information to the attacker during the slow phase.

\(^2\) Optimally this memory is directly connected to the antenna to reduce delays.
The protocol must be designed in such a way that *leaking the set of responses to the attacker will compromise the device’s secrets.*

It is a central assumption in the attack model of the terrorist fraud that the device does not wish to leak those secrets. Hence, for a properly designed protocol the attacker is forced to relay the quick-phase challenges to the legitimate device.

**Exercise 7.1** Would it be easier or harder for deep-sea creatures to design a distance bounding system, given the same level of technology as ours?

### 7.3 The Brands-Chaum protocol

The Brands-Chaum protocol was proposed in 1993. See Fig. 7.1. The cryptographic commitment serves to fix a prover-generated string $m$ such that (a) the verifier cannot read it before it gets ‘opened’ by the prover; (b) the prover cannot lie about $m$, i.e. the action of opening the commitment can yield only $m$ and nothing else.

Many commitment schemes are known in the literature, so we will not be specific.

**Exercise 7.2** (i) Why is the Brands-Chaum protocol secure against the Mafia fraud?
(ii) And why not against the terrorist fraud?
(iii) Can you point out other problems with this protocol?
Figure 7.1: Sketch of the Brands-Chaum distance bounding protocol.
7.4. THE “SWISS KNIFE” PROTOCOL

The Swiss Knife protocol [17], was proposed in 2008. It is an example of a protocol with the following properties: mutual authentication, resistance against mafia and terrorist fraud, error resilience, low computation complexity, and privacy.

The steps are shown in Fig. 7.2.

Exercise 7.3 Why is the Swiss Knife protocol secure against the Mafia fraud? And why against the terrorist fraud?
Exercise 7.4  Consider the terrorist fraud. The legitimate device shares $R^0$ or $R^1$ with the attacker (but not both). Compute the probability of false acceptance, assuming that there is no noise and the verifier tolerates no noise.

Exercise 7.5  In what sense does the Swiss Knife protocol preserve the privacy of the prover? (Please mention two privacy properties.)

7.5 Analog challenge-response processing

Even with state-of-the-art hardware, conversion of the analog challenge signal into a digital bit and then transforming the response bit into an analog signal takes more than 170 nanoseconds (which corresponds to twice traversing 25.5 m). For some applications this is far too much. Hence, the best approach is to process the analog challenge signal into an analog response signal without doing any challenge-dependent digital operations! The incoming signal has to be manipulated in the analog domain, in a way that depends on the bit that has to be transmitted.

The first implementation of this idea was done by Rasmussen and Čapkun in 2010 [27]. It was extended in 2012 [26] in order to be resilient against Terrorist Fraud attacks. The extended system is based on the Swiss Knife protocol, but with a modified challenge-response system in the rapid phase and a slight modification of the $R$ register content. We present a slightly simplified version.

- The prover commits to a mask $M \in \{0, 1\}^{2m}$ which he XORs over the Swiss Knife $R$ register. Other than that, the crypto is exactly as in the Swiss Knife protocol.

- The verifier’s challenge bit is encoded in the frequency of the radio signal: frequency $\omega_0$ represents a ‘0’, and $\omega_1$ represents a ‘1’. The challenge signal in round $i$ as a function of time is denoted as $c_i(t)$. It arrives at the prover with possible modifications due to noise etc., where it is received as $c'_i(t)$.

- Apart from the challenge bit, the challenge signal $c_i(t)$ contains an unpredictable wave shape, e.g. random phase jumps and amplitude modulations. This shape passes through the prover’s circuitry practically unmodified, and gives the verifier a way to see how far away the ‘reflection’ of this signal occurred.

- The prover device has an oscillator running at frequency $\omega_\Delta$, with $\omega_\Delta$ far smaller than $\omega_0$ and $\omega_1$. The incoming challenge signal $c'_i(t)$ is multiplied with the oscillator’s signal. The result is a wave whose frequency components are either $\omega_0 + \omega_\Delta$ and $\omega_0 - \omega_\Delta$ (if the challenge was ‘0’) or $\omega_1 + \omega_\Delta$ and $\omega_1 - \omega_\Delta$ (if the challenge was ‘1’). This follows from basic trigonometry,

$$\begin{align*}
\cos \omega t \cdot \cos \omega_\Delta t &= \frac{1}{2} \cos(\omega + \omega_\Delta)t + \frac{1}{2} \cos(\omega - \omega_\Delta)t \\
\cos \omega t \cdot \sin \omega_\Delta t &= \frac{1}{2} \sin(\omega + \omega_\Delta)t - \frac{1}{2} \sin(\omega - \omega_\Delta)t.
\end{align*}$$  \hfill (7.3)

Then a number of low pass and high pass filters is applied, as shown in Fig. 7.3. Just after the multiplication, two versions of the signal are created: one containing only low frequencies (shown as the upper path), and one with only the high frequencies (lower path). If $c_i(t)$ has not been manipulated with by some adversary, then only one of these two paths actually carries a noticeable signal.

Then each path is split again into two: the upper path is split into a version containing only frequencies lower than $\omega_0$ and a version containing only frequencies higher than $\omega_0$. In the lower path the splitting occurs with a filter that selects frequencies lower than $\omega_1$ and a filter selecting frequencies higher than $\omega_1$. The four output frequencies (‘channels’) shown in Fig. 7.3 are the only possible existing frequencies if $c'_i(t)$ is a compliant signal.

- The antenna is wired to the four channels as follows, depending on the content of the $R$ register: it gets connected to the sum of the $\omega_0 + (2R^0_i - 1)\omega_\Delta$ channel and the $\omega_1 + (2R^1_i - 1)\omega_\Delta$
channel. From the response frequency the verifier learns $R_0^i$ (if the challenge bit was ‘0’) or $R_1^i$ (if the challenge was ‘1’).

**Remark 1:** This $R$-dependent switching does not depend on the challenge and hence can be done using digital electronics.

**Remark 2:** If a malicious verifier makes a signal $c(t)$ that contains both frequencies $\omega_0$ and $\omega_1$, he learns both $R_0^i$ and $R_1^i$. (Remember that these are masked versions of the Swiss Knife’s $R$ registers, and hence meaningless until the mask is opened.)

- In each round, circuitry in the upper and lower path measures how much energy has been received in that path. At the end of the rapid phase (or, alternatively, as soon as digital processing allows it) the prover checks if all energy is properly concentrated into one path per round. If not, the prover decides that the verifier is cheating and aborts the protocol.

- If the prover device decides that the verifier is not cheating, it opens the commitment and reveals the mask $M$ to the verifier. This allows the verifier to unmask the $R$ register and proceed with the rest of the Swiss Knife protocol.

Figure 7.3: Response circuitry in the prover device. The $\otimes$ is a multiplier. The $\Box$ components are frequency filters (either high-pass or low-pass). Shown are the frequencies selected by the filters in case the incoming signal is compliant with the protocol.

With the setup of [27] the processing delay was spectacularly reduced to 1ns (15cm distance between the prover and verifier).
Chapter 8

Quantum physics

8.1 The physics of very small actions

[Section 8.1 is not part of the exam.]

The classical physics that everybody knows from high school, such as electromagnetism and Newtonian mechanics, fails to accurately describe the microscopic world, e.g. superconductivity, superfluid helium, lasers, Bose-Einstein condensation, interactions between light and atoms, chemical reactions, nuclear forces, and a long list of other striking phenomena. Since around 1900 these phenomena have been under intense scrutiny of physicists. As a result of this effort a comprehensive theory has been developed, known as ‘quantum physics’ (or ‘quantum mechanics’), which agrees with experiments with extraordinary accuracy.

Although the behaviour of quantum systems seems to be radically different from what happens in the classical (macroscopic) world, the physical laws governing the dynamics are almost exactly the same! All the classical concepts of energy, momentum, angular momentum, Hamiltonians, Lagrangians, inertia, fields, potentials etc. are essentially the same in quantum physics. The most important difference has to do with the nature of measurements and probability:

1. The outcome of an experiment cannot (in general) be predicted with 100% certainty, even if starting conditions are known exactly. Instead, only probabilities of possible outcomes can be given. This statement has nothing to do with inaccuracies in measurement equipment or lack of knowledge. Information about outcomes of future experiments does not physically exist.

2. The most concise formulation of quantum physics (and at the same time classical physics as well!) is as follows. Given an initial state \( \varphi_0 \) at time \( t = 0 \), the probability that it evolves to an end state \( \varphi_t \) in time \( t \) is given by \( |A_t|^2 \), where

\[
A_t = \int_{\varphi_0}^{\varphi_t} D\varphi \, e^{iS[\varphi]/\hbar}. \tag{8.1}
\]

\( A_t \) is called the amplitude. The \( \varphi \) abstractly denotes the degrees of freedom in the system, e.g. all particle locations and all field strengths. The integral \( \int D\varphi \) is called a path integral. It is a summation over all histories (‘paths’) of the system between \( \varphi_0 \) at time \( t = 0 \) and \( \varphi_t \) at time \( t \) (including completely crazy ones). The \( S[\varphi] \) is the action of the history.\(^1\) \( \hbar \) is Planck’s constant, \( 1.05 \cdot 10^{-34} \) Js.

The path integral expression (8.1) generates all the equations of motion in quantum physics as well as classical physics. When the action is much larger than \( \hbar \), then the complex phases in the integral change vary quickly and have a tendency to cancel each other; there is an overwhelming

\(^1\)In classical mechanics, the action of a set of particles is given by \( \int (E_{\text{kinetic}} - E_{\text{potential}}) dt \). The action of the free electromagnetic field is \( \int (E^2 - c^2 B^2) d^3x dt \).
contribution to $A_t$ from a single path that satisfies $\delta S/\delta \varphi = 0$, precisely the Euler-Lagrange equation that generates all the equations of motion in classical physics. (This is notation from variational calculus, and will not be explained here.)

Some simple but profound rules of thumb directly follow from (8.1).

- If the action of a system is much larger than $\hbar$ then it will follow the rules of classical physics. If $S$ is of the order of $\hbar$ or smaller, the laws of quantum physics apply.
- When degrees of freedom are independent of each other, say $\chi$ and $\omega$, then the path integral factorizes into $\int D\chi \exp(iS[\chi]/\hbar) \cdot \int D\omega \exp(iS[\omega]/\hbar)$ and the above rule of thumb for determining whether there is quantum behaviour applies to the integrals separately.
- If you want to observe quantum phenomena, then the system should not interact strongly with a macroscopic measurement apparatus during the evolution from $\varphi_0$ to $\varphi_t$, otherwise the action grows far beyond $\hbar$.
- Unless a quantum system is isolated from the rest of the world, its interactions with the surroundings will quickly increase the action to macroscopic values. This is known as decoherence.

The first two points allow us to understand why some phenomena involve quantum physics and others do not. Nuclear and chemical processes are quantum because they involve small amounts of matter and short time scales. For the center of mass of a moving brick, the action is very large and consequently it behaves classically. However, the electron orbits of all the atoms in the brick are mostly independent of each other; they each have a small action and obey quantum rules. Photons that move freely through space are completely isolated, hence their behaviour is quantum. Classical optics is in fact room-temperature quantum physics.

**Exercise 8.1** The hydrogen atom consists of a proton and an electron. The radius of the electron’s orbit is $5.29 \cdot 10^{-11}$ m. The electrostatic force between two charged bodies is $q_1 q_2/(4\pi \varepsilon_0 r^2)$, where $q_1, q_2$ are their respective charges, $r$ the distance between them, and $\varepsilon_0 = 8.85 \cdot 10^{-12}$ C$^2$/Jm is the electric constant. Why do we expect to need quantum physics instead of classical physics? (Hint: estimate the action of a single motion of the electron around the proton. You can assume a circular orbit, i.e. the centrifugal force $m v^2/r$ exactly balances the electric attraction.)

### 8.2 Linear algebra refresher

**Exercise 8.2** What are the transpose, complex conjugate and Hermitian conjugate of the complex vector $(a)_b$?

**Exercise 8.3** Give the inner product (also called dot product) of the complex vectors $(a)_b$ and $(c)_d$. What is the norm of $(a)_b$?

**Exercise 8.4** Give the transpose, complex conjugate, Hermitian conjugate, trace, determinant, and the inverse of the complex matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

**Exercise 8.5** Give the definition of a symmetric matrix, a Hermitian matrix and a unitary matrix.

**Exercise 8.6** Compute the following multiplications:

(a) $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

(b) $(e^{*f^*}) (\begin{pmatrix} a \\ b \end{pmatrix})$

(c) $(\begin{pmatrix} a \\ b \end{pmatrix}) (e^{*f^*})$

(d) $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} (\begin{pmatrix} a \\ b \end{pmatrix})$
Exercise 8.7 Find the eigenvectors and eigenvalues of the following matrices:

(a) \[
\begin{pmatrix}
p & 0 & 0 \\
0 & q & 0 \\
0 & 0 & r
\end{pmatrix}
\]

(b) \[
\begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix}
\]

(c) \[
\begin{pmatrix}
0 & 1 \\
-1 & 0
\end{pmatrix}
\]

Exercise 8.8 Express the vectors \[
\begin{pmatrix}
1 \\
0
\end{pmatrix},
\begin{pmatrix}
0 \\
1
\end{pmatrix},
\begin{pmatrix}
p \\
q
\end{pmatrix}
\]
in terms of the basis \[b_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix},
\quad b_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \].

8.3 Dirac notation

With his rectilinear logic, Dirac named each part of the bracket after its first and last three letters, bra and ket, new words that took several years to reach the dictionaries, leaving thousands of non-English speaking physicists wondering why a mathematical symbol in quantum mechanics had been named after an item of lingerie. They were not the only ones to be flummoxed. A decade later, after an evening meal in St John’s, Dirac was listening to dons reflecting on the pleasures of coining a new word, and, during a lull in the conversation, piped up with four words: ‘I invented the bra’.

– From The strangest man by G. Farmelo

Almost all of quantum physics can be derived from (8.1), but in fact the path integral formulation is most suited for the study of collisions and many-particle thermodynamics. We will use the more traditional formulation of quantum physics in terms of ‘wave functions’, Hilbert spaces and operators.

Quantum states are represented as vectors in a Hilbert space. We adopt the usual ‘bra’ and ‘ket’ notation; \(|\psi\rangle\) stands for a quantum state labelled by some description \(\psi\) which summarizes all the knowable information about the state. The Hermitian conjugate is denoted as \(\langle\psi|\). (Think of it as the operator ‘take the inner product with \(\psi\).’) The notation for the inner product between two states is \(\langle\psi_1|\psi_2\rangle\). We will only consider states satisfying \(\langle\psi|\psi\rangle = 1\), so-called normalized states.\(^2\)

Real-valued observables are represented by Hermitian operators acting on the Hilbert space. The \(j\)’th eigenvalue of an observable \(X\) is denoted as \(x_j\), and the corresponding eigenvector as \(|x_j\rangle\). We have \(X|x_j\rangle = x_j|x_j\rangle\). The scalars \(\langle e_i|X|e_j\rangle\), for some basis \(e\), are called the matrix elements of \(X\). The eigenvectors of any Hermitian operator \(X\) form an orthonormal basis of the Hilbert space, i.e. \(\langle x_a|x_b\rangle = \delta_{ab}\). The completeness of this basis (in a finite Hilbert space of dimension \(n\)) is expressed as

\[
\sum_{j=1}^{n} |x_j\rangle\langle x_j| = 1. \tag{8.2}
\]

We will often write \([n]\) for the set \(\{1, 2, \ldots, n\}\). The operator \(X\) can be written as

\[
X = \sum_{j=1}^{n} x_j|x_j\rangle\langle x_j|. \tag{8.3}
\]

Any state \(\psi\) can be expressed in terms of an orthonormal basis,

\[
|\psi\rangle = \sum_{j=1}^{n} c_j|x_j\rangle \quad ; \quad c_j = \langle x_j|\psi\rangle. \tag{8.4}
\]

\(^2\)The states \(|\psi\rangle\) and \(e^{i\chi}|\psi\rangle\) (where \(\chi \in \mathbb{R}\) is some phase angle) describe exactly the same physical state.
Note that $c_j \in \mathbb{C}$ and $\sum_{j=1}^{n} |c_j|^2 = 1$. Measurement of $X$ collapses the state onto one of the eigenvectors (or eigenspaces) of $X$, and yields the corresponding eigenvalue as the measurement result. When a measurement of $X$ is performed on a state $|\psi\rangle$, the probability that $|\psi\rangle$ collapses to the eigenvector $|x_j\rangle$ is given by $|\langle x_j | \psi \rangle|^2$.

In general, a measurement destroys state information.

Note that the outcome of a measurement really is unpredictable. Physical experiments have confirmed that there exists no extra information anywhere in nature adding to the information present in a quantum state $|\psi\rangle$. In fact, quantum measurements are the only source of true randomness known to us!

When no measurements are done, the time evolution of a quantum system can be represented as a unitary operator acting on the starting state. A unitary operator $U$ is defined as one that satisfies $UU^\dagger = 1$ and $U^\dagger U = 1$. It has the special property that the norm of a vector is preserved, i.e. if we have $|\psi'\rangle = U|\psi\rangle$, then the norm is

$$\langle \psi' | \psi' \rangle = \langle \psi | U^\dagger U | \psi \rangle = \langle \psi | \psi \rangle. \quad (8.5)$$

This property is required for the conservation of probability.

### 8.4 Electron spin

Electrons have an internal degree of freedom called spin. It is a form of angular momentum. You can think of it as a tiny dipole magnet shrunk into a point of zero size. It turns out that the electron spin can be described as a state in a two-dimensional Hilbert space. Although the electron spin is a form of angular momentum, you from one to the other, so there are no states in between!

The operators describing spin measurement (let’s call them $\sigma_x, \sigma_y, \sigma_z$ in the Cartesian basis) have to be Hermitian $2 \times 2$ matrices which satisfy the angular momentum algebra,

$$\sigma_x \sigma_y - \sigma_y \sigma_x = 2i \sigma_z, \quad \sigma_y \sigma_z - \sigma_z \sigma_y = 2i \sigma_x, \quad \sigma_z \sigma_x - \sigma_x \sigma_z = 2i \sigma_y. \quad (8.6)$$

There are many ways to construct such matrices. However, it is usual to work in a basis which is diagonalized w.r.t. the $z$ component, i.e. the state $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ denotes a spin fully in the $z$ direction, and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ in the $-z$ direction. This completely fixes the matrices,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (8.7)$$

These are called the Pauli matrices. Their eigenvalues are $\pm 1$. Further properties of these matrices are $\sigma_i^2 = 1$ and

$$\sigma_x \sigma_y = i \sigma_z, \quad \sigma_y \sigma_z = i \sigma_x, \quad \sigma_z \sigma_x = i \sigma_y. \quad (8.8)$$

Sometimes we will also write $\sigma_1, \sigma_2, \sigma_3$ for $\sigma_x, \sigma_y, \sigma_z$. Consider a unit vector $\hat{n} \in \mathbb{R}^3$ characterized by angles $\theta, \varphi$: $\hat{n} = (\sin \theta \cos \varphi, \sin \theta \sin \varphi, \cos \theta)$. The operator for measuring the component of

---

3 The angular momentum operators are given by $\frac{1}{2} \hbar \sigma_i$. We will ignore the factor $\hbar/2$ from this point on.
8.4. ELECTRON SPIN

Spin in the \( \hat{n} \)-direction is

\[
\hat{\sigma}_n = \hat{n}_x \sigma_x + \hat{n}_y \sigma_y + \hat{n}_z \sigma_z = \begin{pmatrix}
\cos \theta & e^{-i\varphi} \sin \theta \\
e^{i\varphi} \sin \theta & -\cos \theta
\end{pmatrix}.
\] (8.9)

The eigenvector of \( \sigma_n \) with eigenvalue 1 is given by (up to a phase factor)

\[
|\hat{n}\rangle = \begin{pmatrix}
\cos \theta/2 \\
e^{i\varphi} \sin \theta/2
\end{pmatrix}.
\] (8.10)

If a measurement of \( \sigma_n \) is done on the state \(|\hat{n}\rangle\) then the outcome is +1 with certainty. If a measurement of \( \sigma_n \) is done, with angle \( \alpha \) between \( \hat{n}' \) and \( \hat{n} \), then the outcome will be +1 with probability \((\cos \frac{\alpha}{2})^2\) and -1 with probability \((\sin \frac{\alpha}{2})^2\).

Note that everything there is to know about the spin state can be summarized in two complex numbers, which is equivalent to four real numbers. In fact, only two real numbers \( \theta \) and \( \varphi \) are sufficient: one degree of freedom is lost because of the constraint \( \langle \psi | \psi \rangle = 1 \), and one is lost because any state multiplied by a phase factor \( e^{i\kappa} \) (with \( \kappa \in \mathbb{R} \)) represents exactly the same physical state: \( e^{i\kappa} |\psi\rangle \) is physically exactly the same as \(|\psi\rangle\).

![The Bloch sphere](image)

**Figure 8.1:** The Bloch sphere, graphically showing the spin direction of the state \( |\psi\rangle = \begin{pmatrix} \cos \theta/2 \\ e^{i\varphi} \sin \theta/2 \end{pmatrix} \). The ‘+z’ state is usually interpreted as the logical ‘0’ state, and the ‘–z’ as the logical ‘1’.

---

**Exercise 8.9** Find the eigenvalues and eigenvectors of \( \sigma_x \) and \( \sigma_y \).

**Exercise 8.10** The electron is in an eigenstate of \( \sigma_x \). A measurement of \( \sigma_x \) is done. What are the probabilities of outcomes +1 and -1? What if the starting state was an eigenstate of \( \sigma_y \)?

**Exercise 8.11** The electron spin is in initial state \( |\psi\rangle = \begin{pmatrix} \sqrt{1-\alpha^2} \\ i\alpha \exp(\text{i}b) \end{pmatrix} \). A measurement is done of \( \sigma_z \). What is the probability of the result -1? If -1 is measured, what is the spin state after the measurement?

**Exercise 8.12** The angular momentum operators are given by \( J_i = \frac{\hbar}{2} \sigma_i \). The ladder operators for the z-direction are defined as \( J_\pm = J_x \pm iJ_y \). Show (without using the explicit form of the eigenstates of \( \sigma_z \)) that \( J_+ \) increases the z-component of the angular momentum by \( \hbar \). Hint: first compute the so-called commutator \( [J_z, J_+] = J_z J_+ - J_+ J_z \).
8.5 Polarization of light

8.5.1 Classical polarization

Light is a self-propagating oscillation of the electromagnetic field in empty space, with velocity $c = 3.00 \cdot 10^8$ m/s. The electric and magnetic field are both perpendicular to the direction of motion, and they are perpendicular to each other. Light is said to be linearly polarized when the electric field as a function of time varies in only one dimension, e.g. the light moves in the $z$-direction and the electric field is $\mathbf{E}(t) \propto \hat{x} E \cos \omega t$, where $\hat{x}$ is the unit vector in the $x$-direction, and $\omega$ is the angular frequency of the light. Light is said to be circularly polarized when the electric field moves in a circle. Elliptic polarization is also possible, and is in fact the most general case.

Consider light moving in the $z$-direction. For a given value of the time average $\langle \hat{E}^2 \rangle$ at a fixed location in space, the polarization is characterized by two angles $\beta \in (-\pi/2, \pi/2)$, $\chi \in [-\pi/4, \pi/4]$. The $\beta$ is the angle from the $x$-axis to the long axis of the ellipse. The $\chi$ determines the ellipticity: $\chi = 0$ means linear, $\chi = \pi/4$ means right-handed circular, and $\chi = -\pi/4$ means left-handed circular. Let $a$ and $b$ denote the semimajor and semiminor axis of the ellipse. Then $\tan \chi = b/a$.

The electric field as a function of time is given by

$$\mathbf{E}(t) = \begin{pmatrix} E_x(t) \\ E_y(t) \end{pmatrix} = \sqrt{2\langle \mathbf{E}^2 \rangle} \begin{pmatrix} \cos \beta & -\sin \beta \\ \sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} \cos \chi \cos \omega t \\ \sin \chi \sin \omega t \end{pmatrix}$$

$$= \frac{1}{\sqrt{\langle \mathbf{E}^2 \rangle}} \begin{pmatrix} \sqrt{1+\cos 2\chi \cos 2\beta} \cos(\omega t + \arctan(\tan \chi \tan \beta)) \\ \sqrt{1-\cos 2\chi \cos 2\beta} \cos(\omega t - \arctan(\tan \chi \tan \beta)) \end{pmatrix}.$$  

(8.11)

This expression can be interpreted as a combination of two linear polarizations, $E_x(t)$ and $E_y(t)$, that are out of phase.

8.5.2 Polarization of a single photon

Around 1905 it was realized (Planck, Einstein) that light is not a continuous thing. The smallest amount of light of frequency $\omega$ that can exist is a package with energy $\hbar \omega$. Such a ‘quantum of light’ is called a photon. Just like macroscopic light, a photon has polarization. The Hilbert space of the polarization is two-dimensional. That is not surprising, as we have already seen in (8.11) that any type of polarization can be described by two parameters $(\beta$ and $\chi$) and can be built up from horizontal and vertical polarization. As a basis of the Hilbert space one can choose the orthogonal states $|\leftrightarrow\rangle$ and $|\uparrow\rangle$.

$|\leftrightarrow\rangle$ corresponds to $\hat{E}(t) \propto \hat{\epsilon}_x \exp[i\omega t]$.

$|\uparrow\rangle$ corresponds to $\hat{E}(t) \propto \hat{\epsilon}_y \exp[i\omega t]$.

$\langle\leftrightarrow|\uparrow\rangle = 0$.

---

For electromagnetic waves, the frequency $f$ and wavelength $\lambda$ are related according to $c = \lambda f$. Physicists more often use angular frequency $\omega := 2\pi f$ and wave number $k := 2\pi/\lambda$. The relation between $\omega$ and $k$ is $\omega = ck$. The ‘wave vector’ $\mathbf{k}$ is defined as a vector of length $k$ pointing in the direction in which the light travels. Working with $\omega$ and $k$ instead of $f$ and $\lambda$ simplifies the notation a lot. For instance, the phase of a plane wave at location $\mathbf{x}$ and time $t$ is expressed as $\exp(i\mathbf{k} \cdot \mathbf{x} - \omega t)$. The energy and momentum of a photon are $\hbar \omega$ and $\hbar \mathbf{k}$ respectively.
A general polarization state has the form

\[ |\psi\rangle = c_h|\leftrightarrow\rangle + c_v|\uparrow\downarrow\rangle, \]

(8.12)

with \(c_h, c_v \in \mathbb{C}\) and \(|c_h|^2 + |c_v|^2 = 1\). If \(c_h\) and \(c_v\) have the same phase (by that we mean \(c_v/c_h \in \mathbb{R}\)) then the polarization is linear (\(\chi = 0\)), in the direction \(\beta = \arctg(c_v/c_h)\). A photon with state \(|\psi\rangle\) can be said to be in a quantum superposition of horizontal and vertical polarization; this means it is in both states simultaneously.

A two-level quantum system is usually described using states of the form \(e^{i\phi} \sin \theta/2\) where \(\theta\) and \(\varphi\) have a clear meaning in terms of spin (see the Bloch sphere, Fig. 8.1). From (8.11) we obtain the relation between the Bloch sphere angles on the one hand and the polarization parameters on the other hand,

\[
\begin{align*}
\tan 2\beta &= \cos \varphi \tan \theta ; & \sin 2\chi &= \sin \varphi \sin \theta \\
\cos \theta &= \cos 2\chi \cos 2\beta ; & \tan \varphi &= \frac{\tan 2\chi}{\sin 2\beta}.
\end{align*}
\]

(8.13)

Note that \(\cos \frac{\theta}{2} = \sqrt{1 + \cos 2\chi \cos 2\beta} / \sqrt{2}\) and \(\sin \frac{\theta}{2} = \sqrt{1 - \cos 2\chi \cos 2\beta} / \sqrt{2}\), which indeed matches the amplitudes in (8.11). Subtraction of the two phases in (8.11), \(\tan \Phi_1 := \tan \chi \tan \beta\) and \(\tan \Phi_2 := -\tan \chi / \tan \beta\) yields

\[
\tan (\Phi_1 - \Phi_2) = \frac{\tan \Phi_1 - \tan \Phi_2}{1 + \tan \Phi_1 \tan \Phi_2} = \frac{\tan \chi}{1 - (\tan \chi)^2} \frac{\sin \beta}{\cos \beta} + \frac{\cos \beta}{\sin \beta} = \frac{\tan 2\chi}{\sin 2\beta} = \tan \varphi.
\]

(8.14)

Figure 8.3: The Bloch sphere for photon polarization. Six special states are shown: Horizontal, Vertical, Diagonal (45°), Antidiagonal (−45°), Right circular, Left circular. All linearly polarized states lie on the great circle through H,D,V,A. Opposite points on this circle have orthogonal polarization directions. Any polarization that is not linear or circular is elliptic. The hemisphere centered on R has right-handed polarization; the hemisphere centered on L is left-handed. The hemisphere centered on H has semimajor axis closer to the x-axis than to the y-axis. In the hemisphere centered on V it is the other way round.

Exercise 8.13 Show that the 2nd line in (8.11) indeed follows from the 1st line.

Consider linearly polarized light with orientation \(\beta \in (0, \pi/2)\), falling on a filter that lets through only horizontally polarized light. In the classical case, a fraction \(\cos^2 \beta\) of the light is transmitted and the rest is absorbed.
In the single-photon case, we have a polarization state $|\beta\rangle = \cos \beta |\leftrightarrow\rangle + \sin \beta |\downarrow\rangle$. The filter corresponds to a measurement of the projection operator $P_{\text{hor}} := |\leftrightarrow\rangle\langle\leftrightarrow|$ which selects only the horizontal polarization. The eigenstates of $P_{\text{hor}}$ are $|\leftrightarrow\rangle$ and $|\downarrow\rangle$, with eigenvalues 1 and 0 respectively. When we apply the rules of Section 8.3, we see that the measurement collapses the state to $|\leftrightarrow\rangle$ with probability $(\cos \beta)^2$, and to $|\downarrow\rangle$ with probability $(\sin \beta)^2$. Hence the photon passes through the filter with probability $\cos^2 \beta$ and gets absorbed with probability $\sin^2 \beta$. It is important to remark that it is impossible to predict which of these two events will occur! This is not a limitation due to imprecise knowledge of the initial conditions. It is a fundamental fact of quantum physics: even a perfectly detailed description of the initial conditions does not contain any information about the outcome of the experiment. The outcome is a stochastic variable generated ‘out of nowhere’. Quantum systems are perfect (one could in fact say the only) true random number generators.

There is a neat correspondence between the microscopic (single-photon) and macroscopic case. Consider one $\mu$W of visible light ($\lambda = 500\text{nm}$) hitting the filter for one microsecond; this corresponds to some $N_\varphi = 10^7$ photons. Each photon individually gets transmitted with probability $(\cos \beta)^2$ . The total number of transmitted photons follows a binomial distribution. The expectation value of the number of transmitted photons is $N_\varphi \cos^2 \beta$, i.e. on average a fraction $\cos^2 \beta$ passes, just as in the classical case. The variance (\(\sigma\)) is $\sqrt{N_\varphi \cos^2 \beta (1 - \cos^2 \beta)} = \frac{1}{2} \sin 2\beta \sqrt{N_\varphi} \ll N_\varphi$. We see that even for such small amounts of light, there are so many photons involved that the statistical deviations due to quantum effects are hardly noticeable.

### 8.6 Qubits

Let us abstract away from the physics for the moment. Given that there exist physical systems with a two-dimensional Hilbert space (such as the electron spin), it should be possible to treat the states as ‘bits’ in some sense. Operators such as $\sigma_z$ have two distinct eigenstates, which we could interpret as a logical ‘0’ and ‘1’. We could for instance define $|0\rangle = |+z\rangle$ and $|1\rangle = |–z\rangle$ in the case of electron spin. Such an interpretation of a two-state quantum system is called a quantum bit, or qubit.

A qubit has some unusual properties compared to an ordinary bit. It can be prepared to be in a combination of 0 and 1, called a superposition. We could for instance put a qubit in the state

$$|\psi\rangle = a|0\rangle + b|1\rangle,$$

(8.15)

with $|a|^2 + |b|^2 = 1$. A computation on a single qubit can be seen as a unitary operator $U$ acting on the state,

$$U|\psi\rangle = a \cdot U|0\rangle + b \cdot U|1\rangle.$$

(8.16)

Clearly the computation is done on the logical 0 and 1 simultaneously by the same physical operation! This is sometimes called ‘quantum parallel computation’. Don’t get your hopes up though: it is almost impossible to capitalize on this form of parallelism. Speeding up computations turns out to be impossible except in a small number of very special cases (see Chapter 10). The reason lies in the readout of qubits. The only kind of thing that can be measured is projection operators such as $|0\rangle\langle0|$ and $|1\rangle\langle1|$. The only thing you learn from the outcome is ‘oh, apparently the state had a nonzero component in the direction of the outcome’. For instance, in (8.15) you can put an infinite amount of information in the continuum variables $a$ and $b$. But if you measure $|1\rangle\langle1|$ on the state $U|\psi\rangle$ and get out +1, then the only thing you can conclude is that $U|\psi\rangle$ is not equal to $|0\rangle$. In order to learn more, you have to estimate $\langle1|U|\psi\rangle$, which takes multiple repetitions of the same experiment... and gone is the advantage of quantum parallelism.
8.7 Combining multiple quantum systems

8.7.1 Tensor products

Qubits are of no use if they cannot be combined. Mathematically speaking, describing two quantum systems together is done by taking the tensor product. If we have two Hilbert spaces $\mathcal{H}_1, \mathcal{H}_2$, then the combined Hilbert space is $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$. If we have two independent particles (meaning that they do not feel each other) with states $|\omega\rangle_1 \in \mathcal{H}_1$ and $|\chi\rangle_2 \in \mathcal{H}_2$, then their combined state is the tensor product $|\omega\rangle_1 \otimes |\chi\rangle_2 \in \mathcal{H}$. The tensor product of vectors is computed as follows,

$$
\begin{pmatrix}
\alpha \\
\beta
\end{pmatrix} \otimes 
\begin{pmatrix}
\gamma \\
\delta
\end{pmatrix} = 
\begin{pmatrix}
\alpha \gamma \\
\alpha \delta \\
\beta \gamma \\
\beta \delta
\end{pmatrix}.
$$

(8.17)

However, if the particles do feel each other, then in general their combined state will not be a tensor product of states in $\mathcal{H}_1, \mathcal{H}_2$; instead it is written as

$$
|\psi\rangle = \sum_{a=1}^{\dim\mathcal{H}_1} \sum_{b=1}^{\dim\mathcal{H}_2} C_{ab} |a\rangle_1 \otimes |b\rangle_2.
$$

(8.18)

(The subscripts 1 and 2 are often omitted if the meaning is clear.) The $C_{ab}$ are complex coefficients satisfying $\sum_{ab} |C_{ab}|^2 = 1$. The states $|a\rangle$ form a basis of $\mathcal{H}_1$, and $|b\rangle$ of $\mathcal{H}_2$. The dimension of the combined Hilbert space in (8.18) is $\dim\mathcal{H} = \dim\mathcal{H}_1 \cdot \dim\mathcal{H}_2$. A state of the form (8.18) is called an entangled state if it cannot be factorized into the form $|\cdot\rangle_1 \otimes |\cdot\rangle_2$.

Operators from different Hilbert spaces can be combined by taking their tensor product. The meaning of such a construction is the multiplication of their outcomes. For instance,

$$(X \otimes Y) |x_i\rangle \otimes |y_j\rangle = (X|x_i\rangle) \otimes (Y|y_j\rangle) = x_i y_j |x_i\rangle \otimes |y_j\rangle.
$$

(8.19)

Often one is interested only in a measurement within one of the subspaces. For instance, acting with $Y$ on particle 2 is formally written as $(1 \otimes Y)|\psi\rangle$.

Let’s return to the qubits. The combined state of two qubits can be written as

$$
|\psi\rangle = \sum_{b_0=0}^{1} \sum_{b_1=0}^{1} C_{b_1,b_0} |b_1\rangle \otimes |b_0\rangle,
$$

(8.20)

with $|C_{00}|^2 + |C_{01}|^2 + |C_{10}|^2 + |C_{11}|^2 = 1$. For qubits the tensor product symbol $\otimes$ is often omitted, and notation like $|0\rangle|1\rangle$ will then occur for $|0\rangle \otimes |1\rangle$, or even more compactly $|01\rangle$. If we choose to write $|0\rangle = (1, 0)^T$, then the four logical states are given by the following four-dimensional vectors,

$$
\begin{pmatrix}
1 \\
0 \\
0 \\
0
\end{pmatrix} \quad |01\rangle = 
\begin{pmatrix}
0 \\
1 \\
0 \\
0
\end{pmatrix} \quad |10\rangle = 
\begin{pmatrix}
0 \\
0 \\
1 \\
0
\end{pmatrix} \quad |11\rangle = 
\begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}.
$$

(8.21)

The expression (8.20) simply begs for re-interpretation in terms of the binary representation of the integers $\{0, 1, 2, 3\}$. Let $\mathcal{H}_1$ denote the Hilbert space of a single qubit. We define $|x\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_1$, with $x \in \{0, 1, 2, 3\}$, as the tensor product of the bits making up $x$, i.e.

$$
\begin{align*}
|0\rangle &= |00\rangle \\
|1\rangle &= |01\rangle \\
|2\rangle &= |10\rangle \\
|3\rangle &= |11\rangle.
\end{align*}
$$

(8.22)
Eq. (8.20) can then be rewritten in the more appealing form

$$ |\psi\rangle = \sum_{x=0}^{3} c_x |x\rangle, $$

(8.23)

i.e. a superposition of all the two-bit logical states. Eq. (8.23) is trivially generalized to more than two qubits.

We return to the question of ‘quantum parallel computation’ and its limitations. A general state of an \(n\)-qubit register can be written as

$$ |\psi\rangle = \sum_{x=0}^{2^n-1} c_x |x\rangle. $$

A single unitary operation \(U\) applied to \(|\psi\rangle\) performs \(2^n\) computations at the same time, resulting in a final state

$$ \sum_{x} c_x U|x\rangle, $$

i.e. a superposition of all the \(2^n\) results \(U|x\rangle\). Readout of the logical state of the quantum register (or some other equivalent measurement) will result in a projection onto one of the \(2^n\) basis directions in the Hilbert space. For someone who does not know the computed values (and this is by default the case; why would you wish to do the computation otherwise?) the entropy of this projection event is \(\log_2(2^n) = n\). Hence, measuring the logical state of an \(n\)-qubit register yields at most \(n\) bits of information, the same amount as for a classical register.

### 8.7.2 Measurement on a subsystem

Some words have to be spent on the effect of ‘subsystem’ measurements on entangled states. As before, a measurement of an observable projects the state onto the eigenspace corresponding to the outcome. When there are multiple particles, and only one particle is measured, this projection is less trivial than in the single-particle-case. Consider the following entangled state,

$$ |\chi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|11\rangle, $$

(8.24)

with \(|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1\). If the logical value of the first particle is measured, the probability of outcome ‘1’ is \(|\gamma|^2\). If this outcome is obtained, then the state of the system is projected onto \(|11\rangle\).

On the other hand, if the outcome is ‘0’ then projection occurs onto \((\alpha|00\rangle + \beta|01\rangle) / \sqrt{|\alpha|^2 + |\beta|^2}\).

Why? Because the \(|11\rangle\) term is incompatible with the outcome, and the projected state has to be normalized again. It helps to write \(|\chi\rangle\) as a decomposition in terms of the logical states of particle 1, and take care to explicitly write out the normalization of the state of particle 2,

$$ |\chi\rangle = \sqrt{1-|\gamma|^2} |0\rangle \otimes \frac{\alpha|0\rangle + \beta|1\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}} + \gamma|1\rangle \otimes |1\rangle. $$

(8.25)

**Exercise 8.14** Show that the tensor product in (8.17) results in a normalized state (norm 1).

**Exercise 8.15** Given the state (8.24), the logical value of particle 2 is measured. What are the possible outcomes and their probabilities? What is the state of the two-qubit system after the measurement?

---

5 The corresponding operator is \((0 \cdot |0\rangle\langle 0| + 1 \cdot |1\rangle\langle 1|) \otimes 1\)
8.8 How to exploit all this

In chapter 10 we will see how the qubits described above can serve as the building blocks for quantum computers which can do something useful, like breaking RSA and Diffie-Hellman.

There are other ways to make good use of the peculiarities of quantum evolution and quantum measurements. In particular, the fact that (in general) measurements destroy information can be exploited for cryptographic purposes. There is a famous result called the no-cloning theorem which basically states that it is impossible to copy some unknown state of one particle onto another particle while preserving the state of the first particle.

**Theorem 8.1 (No-cloning theorem)** Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two Hilbert spaces. Let $|\psi\rangle \in \mathcal{H}_1$ and $|e\rangle \in \mathcal{H}_2$, where $e$ is known and $\psi$ is unknown. Then there does not exist a unitary operator $U_e$ acting on $\mathcal{H}_1 \otimes \mathcal{H}_2$ satisfying $U_e |\psi\rangle \otimes |e\rangle = |\psi\rangle \otimes |\psi\rangle$ for all $\psi$.

**Proof:** Assume that such a unitary $U_e$ exists. (It may depend on $e$.) Then for two arbitrary $\psi_1, \psi_2$ it holds that $U_e |\psi_1\rangle |e\rangle = |\psi_1\rangle |\psi_1\rangle$ and $U_e |\psi_2\rangle |e\rangle = |\psi_2\rangle |\psi_2\rangle$. The inner product between $U_e |\psi_1\rangle |e\rangle$ and $U_e |\psi_2\rangle |e\rangle$ is given by

$$\langle e | \langle \psi_2 | U_e^\dagger U_e | \psi_1 \rangle | e \rangle = \langle \psi_2 | \psi_1 \rangle.$$  

(8.26)

(Here we have written $[|\psi_2\rangle |e\rangle]^\dagger = \langle e | \langle \psi_2 |$.) On the other hand, the same inner product is also given by

$$\left(\langle \psi_2 | \psi_2 \rangle \right) \left(|\psi_1\rangle |\psi_1\rangle\right) = \langle \psi_2 | \psi_1 \rangle^2.$$  

(8.27)

Now for arbitrary $\psi_1, \psi_2$ it is not true that $\langle \psi_2 | \psi_1 \rangle^2 = \langle \psi_2 | \psi_1 \rangle$. We conclude that $U_e$ cannot have the stated property. □

**Exercise 8.16** Is it possible to clone a known quantum state?

**Exercise 8.17** Is it possible to clone an unknown quantum state if the number of candidate states is low?

The no-cloning theorem was invoked in a paper [4] where it was realized that a secret single-particles quantum state (known only by the issuer) can be used as an unclonable ticket. Anyone trying to clone the ticket will fail with high probability, while the issuer knows how to verify the ticket with high probability.

A more spectacular use of the no-cloning theorem was proposed in [3], where it was shown how two parties exchanging photons with secret polarization states can create an unconditionally secure shared key. The next chapter is devoted to this trick, variously called ‘quantum key distribution’, ‘quantum key exchange’ or ‘quantum cryptography’.
Chapter 9

Quantum Key Distribution

9.1 The BB84 protocol

In this section we discuss the Quantum Key Distribution protocol as proposed by Bennett and Brassard [3] in 1984. The protocol is very simple, and makes use of single quanta with a two-dimensional internal state, e.g. polarization of single photons as discussed in Section 8.5.2. A lot of progress has been made in this field since 1984, e.g. different physical systems, use of many-particle states, improved proof techniques, and tricks to reduce the effectiveness of eavesdropping. However, BB84 is still the simplest protocol to explain.

9.1.1 Attack model and physical context

Alice and Bob wish to generate a long, information-theoretically secure key. They can talk to each other over an insecure but authenticated channel, i.e. Eve hears everything that is said on this channel, but Alice and Bob know that they are really talking to each other; Eve cannot manipulate their messages. Furthermore, Alice can send single photons to Bob, and Bob has a single-photon detector. Eve is able to intercept Alice’s photons and to send her own photons to Bob. We say that Eve is successful if Alice and Bob agree on a key and Eve learns a substantial part of this key.

Alice and Bob agree on two bases of the Hilbert space,

- **The + basis.** A measurement in the + basis is realized by placing a vertical polarization filter in front of the detector. The corresponding operator is $|\up\rangle\langle\up|$, with eigenstates $|\leftrightarrow\rangle$ (eigenvalue 0) and $|\down\rangle$ (eigenvalue 1).

- **The × basis.** A measurement in the × basis is realized by placing a $\times$ polarization filter in front of the detector. The corresponding operator is $|\times\rangle\langle\times|$, with eigenstates $|\wedge\rangle$ (eigenvalue 0) and $|\vee\rangle$ (eigenvalue 1). In terms of the + basis we can define the × basis vectors as

$$|\wedge\rangle = \frac{|\leftrightarrow\rangle + |\down\rangle}{\sqrt{2}}, \quad |\vee\rangle = \frac{|\leftrightarrow\rangle - |\down\rangle}{\sqrt{2}}.$$  \tag{9.1}

We have $\langle \wedge | \vee \rangle = 0$. Since the two bases are tilted by $45^\circ$ with respect to each other, the expression $(\cos 45^\circ)^2 = 1/2$ appears in the following inner products,

$$|\langle \wedge | \wedge \rangle|^2 = \frac{1}{2}, \quad |\langle \wedge | \vee \rangle|^2 = \frac{1}{2},$$

$$|\langle \vee | \wedge \rangle|^2 = \frac{1}{2}, \quad |\langle \vee | \vee \rangle|^2 = \frac{1}{2}.\tag{9.2}$$

Eve’s measurement and state preparation equipment is assumed to be perfect. Eve is allowed to do any unitary transformation on the photons.

**Exercise 9.1** Verify that $\langle \wedge | \vee \rangle = 0$ and (9.2) indeed hold.
### 9.1.2 The protocol

The protocol is shown in Fig. 9.1.

1. For \( i = 1 \) to \( n \):
   
   (a) Alice picks a random basis \( V_i \) (\(+\) or \( \times \)) and a random bit \( b_i \). She looks up the corresponding state \( \psi \) in Table 9.1. She prepares a photon in state \( |\psi\rangle \) and sends it to Bob.

   (b) Bob chooses a random basis \( V'_i \). He performs a polarization measurement of the photon in this basis\(^1\), obtaining result \( b'_i \).

2. Bob tells Alice (over the classical channel) all his basis choices \( V_1, \ldots, V_n \).

3. Alice selects the rounds where the same basis was chosen: \( E = \{ i : V_i = V'_i \} \). She also picks a random subset of indices \( S \subset E \). She sends \( S \) and \( E \) to Bob.

4. Bob returns \( b'_S \) to Alice.

5. Alice checks if \( b'_S \approx b'_E \). If there are too many bit errors\(^2\), then there must be something wrong (eavesdropping or too much noise on the quantum channel) and she aborts the protocol.

At this point (provided that Alice finds that \( b'_S \) is sufficiently close to \( b'_E \)) Alice and Bob almost have a shared secret: Alice has \( b'_{E \setminus S} \) and Bob has \( b'_{E \setminus S} \). With overwhelming probability the difference between these bit strings is determined by the (known) error rate of the quantum channel. However, some of the differences may have been caused by a small amount of eavesdropping. Hence, there are two things standing in the way of a perfect key: noise, and the fact that the adversary possibly knows some key bits.

The next step in the BB84 protocol is **error correction**. This can be done in many ways. For instance, Alice and Bob both map their bit string to the nearest code word in a code that they agreed on in advance.

Finally they have to apply **privacy amplification** (see Section 3.4) in order to reduce Eve’s possible knowledge about the key.

### 9.1.3 The security of BB84

Let us consider a simple intercept-resend attack. We look at a round where \( B_i = B'_i \) (see Table 9.2). Eve selects a random basis \( B''_i \) and does a measurement in this basis. She obtains a bit \( b''_i \). She prepares a new photon in the \( (B''_i, b''_i) \) state and sends it to Bob. If she correctly guessed the basis (50% probability) then she learns \( b_i \) and her actions go unnoticed. If she guessed wrong (50% probability), then \( \Pr[b''_i = b_i] = 1/2 \) and \( \Pr[b''_i \neq b_i] = 1/2 \). Hence, her overall probability of disturbing a bit is 1/4, and her overall probability of learning \( b_i \) is 3/4. If she eavesdrops on every bit, then the disturbances cause a bit error rate of 1/4 in \( b'_S \), which is noticed in step 5.

\[^1\text{If they chose the same basis, then } b'_i = b_i \text{ with } 100\% \text{ certainty. If not, then } b'_i \text{ is } 0 \text{ or } 1 \text{ with } 50/50 \text{ probability.}\]

\[^2\text{A certain nonzero error rate is tolerated in order to allow for ordinary noise.}\]
9.1. THE BB84 PROTOCOL

<table>
<thead>
<tr>
<th>basis</th>
<th>b</th>
<th>$\psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\times$</td>
<td>0</td>
<td>$\uparrow$</td>
</tr>
<tr>
<td>$\times$</td>
<td>1</td>
<td>$\rightarrow$</td>
</tr>
<tr>
<td>$+$</td>
<td>0</td>
<td>$\leftrightarrow$</td>
</tr>
<tr>
<td>$+$</td>
<td>1</td>
<td>$\uparrow$</td>
</tr>
</tbody>
</table>

Random basis. Random bit b. $\rightarrow$ Random basis. Measure b'.

Keep events with equal basis:
subset E.
Small random set $S \subseteq E$.

Check if $b_S \approx b'_S$.

<table>
<thead>
<tr>
<th>$E, S$</th>
<th>$b'_S$</th>
</tr>
</thead>
</table>

Shared secret $b_{ES} \approx b'_{ES}$.

- Error correction
- Privacy amplification

Figure 9.1: The BB84 protocol.

<table>
<thead>
<tr>
<th>Alice’s state</th>
<th>$\leftrightarrow$</th>
<th>$\leftrightarrow$</th>
<th>$\downarrow$</th>
<th>$\downarrow$</th>
<th>$\leftarrow$</th>
<th>$\rightarrow$</th>
<th>$\uparrow$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eve’s basis</td>
<td>$+$</td>
<td>$\times$</td>
<td>$+$</td>
<td>$\times$</td>
<td>$+$</td>
<td>$\times$</td>
<td>$+$</td>
</tr>
<tr>
<td>Eve’s outcome</td>
<td>$\leftrightarrow$</td>
<td>$\downarrow$ $\leftrightarrow$</td>
<td>$\downarrow$ $\leftrightarrow$</td>
<td>$\leftrightarrow$</td>
<td>$\leftrightarrow$</td>
<td>$\leftrightarrow$ $\leftrightarrow$</td>
<td></td>
</tr>
<tr>
<td>Bob’s basis</td>
<td>$+$</td>
<td>$+$</td>
<td>$+$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>Bob’s outcome</td>
<td>0</td>
<td>0 or 1</td>
<td>1</td>
<td>0 or 1</td>
<td>0 or 1</td>
<td>0</td>
<td>0 or 1</td>
</tr>
</tbody>
</table>

Table 9.2: Possible events occurring when Alice and Bob choose the same basis and Eve does an intercept-resend attack. Where there are two possibilities listed, the probabilities are 1/2. Whenever Eve guesses the correct basis, she does not disturb the photon and learns its state. Whenever she guesses wrong, Bob has a 50% probability of getting the wrong outcome.
There exist more sophisticated attacks, of the intercept-resend type but also on several photons simultaneously. BB84 has been proven secure against all possible attacks. (There are various proof methods. They are too complicated for these lecture notes.) They key is information-theoretically (unconditionally secure). No computational assumptions underlie the security. There is only the assumption that the laws of physics as we know them are correct.

Eve’s knowledge about the key can be made arbitrarily small by increasing the compression ratio of the privacy amplification step. (This means discarding more bits.)

9.1.4 Remarks about quantum key distribution

- Alice and Bob have an authenticated classical channel. In practice this is achieved by starting out with a (short) secret MAC key. But wait a minute ..., isn’t that cheating? The whole point of the exercise was to derive a secret key, and now it turns out that they already start with a secret key! No, it is not cheating: At the beginning they have a short key, and they use the BB84 protocol to make it arbitrarily long.

- After step 5, the situation is very similar to Fuzzy Extractors and the satellite scenario of Chapter 4 (Key agreement from correlated randomness). The main difference is where the correlated random bits come from, and what the properties of the channel are.
Chapter 10
Quantum computers

[The following parts of this chapter will be part of the exam: 10.1 and 10.2.3–10.2.4.]

10.1 Operations on qubits: quantum gates

It is (in theory) possible to build a quantum computer based on the qubits introduced in Section 8.6. The qubits provide the storage. What is needed apart from storage is (i) a method of preparing qubits in a desired logical state; (ii) operations on single qubits independently; (iii) logical operations on multiple qubits; (iv) a readout mechanism.

It has been shown with various physical implementations that a single qubit can be very well controlled.\footnote{For instance, the direction of a single electron spin or nuclear spin can be controlled very precisely by applying magnetic fields of the right direction for the right amount of time.} Hence, state preparation and single-qubit manipulation are easy to achieve. Furthermore, single-qubit readout in the ‘logical’ basis is a simple operation; by applying it to many qubits independently one effectively achieves a multi-qubit readout in the logical basis.

The toughest engineering problem is to come up with the quantum equivalent of logical gates. In order to do something nontrivial involving more than one qubit, the qubits have to ‘feel’ each other. However, at the same time qubits have to be designed such that they preserve their quantum state when they are not being accessed, i.e. they must not feel the outside world.

Let us for the moment assume that the engineering problems have been solved. What kind of operations are possible? Remember that the evolution of a quantum system (without measurements taking place) is described by a unitary operator acting on the state. One of the key properties of unitary operators is that they are invertible: no information is lost, and any event can be reversed. As an immediate consequence, some ordinary logical gates cannot be implemented. Consider for instance the binary NAND gate. It has two binary inputs and one binary output. If the output is 1, then it is not possible to uniquely reconstruct the input from the output. (The input could have been 00 or 01 or 10.) If you want to implement an operation in a quantum computer, you must take care that the information contained in the inputs is preserved. This severely limits the construction of gates. Luckily, with some effort a complete set can be found which is capable of performing the equivalent of all classical logical operations.

Example 10.1 Phase gate. The phase gate acts on a single qubit. It maps $|0\rangle$ to $|0\rangle$, and $|1\rangle$ to $e^{i\varphi}|1\rangle$, with $\varphi \in \mathbb{R}$. The corresponding unitary operator and the matrix representation in the logical basis are

\[
|0\rangle\langle 0| + e^{i\varphi}|1\rangle\langle 1| \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix}.
\] (10.1)

Exercise 10.1 Show that the phase gate operator is unitary.
Exercise 10.2 A qubit has state $e^{i\alpha}|1\rangle$, with $\alpha \in \mathbb{R}$. Then a phase gate operation is applied to the qubit. Is its logical state changed?

Exercise 10.3 A qubit has state $c|0\rangle + d|1\rangle$, with $|c|^2 + |d|^2 = 1$ and $c \neq 0, d \neq 0$.

(a) If the logical state is measured, what are the probabilities of getting the outcomes 0 and 1?

(b) Now the phase gate operation is applied to the qubit. Again give the probabilities for the logical states.

(c) In what sense has the state of the qubit changed?

Example 10.2 Hadamard gate. The Hadamard gate acts on a single qubit. It swaps between the ordinary (logical) basis and the so-called ± basis with basis states $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. It maps $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $\rangle$.

The corresponding unitary operator and the matrix representation in the logical basis are

$$
\frac{|0\rangle + |1\rangle}{\sqrt{2}} = \begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix}.
$$

Exercise 10.4 Show that the Hadamard gate is unitary.

Exercise 10.5 Show that the Hadamard gate maps $|+\rangle$ to $|0\rangle$, and $\rangle$ to $|1\rangle$.

Exercise 10.6 Consider a two-qubit system in a starting state $|00\rangle$. Both qubits are independently subjected to the Hadamard gate operation. Then a measurement is done of the two-qubit logical state (see Section 8.7.1). What are the possible outcomes and their probabilities?

Exercise 10.7 Same as exercise 10.6, but now for 3 qubits.

Example 10.3 Controlled NOT gate (CNOT). The CNOT gate maps an input $(x_1, x_2)$ to an output $(y_1, y_2)$ as follows. $y_1 = x_1$; $y_2 = x_2$ unless $x_1 = 1$, in which case $y_2 = x_2$. In other words, $x_1$ controls whether the NOT operation is applied to $x_2$. The truth table and the corresponding unitary operator (both in Dirac notation and in the logical basis) are shown below.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y_1$</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| $\langle 00|00 \rangle + \langle 01|01 \rangle + |11\rangle\langle 10 | + |10\rangle\langle 11 | = \\
| 1 0 0 0 \\
| 0 1 0 0 \\
| 0 0 0 1 \\
| 0 0 1 0 |

Example 10.4 Controlled U gate. This operation acts on two qubits. The input bit $x_1$ passes unchanged. It controls whether the one-qubit unitary operation $U$ is applied to input bit $x_2$. The corresponding operator is

$$
|00\rangle\langle 00 | + |01\rangle\langle 01 | + \left( |1\rangle \otimes U |0\rangle \right) \langle 10 | + \left( |1\rangle \otimes U |1\rangle \right) \langle 11 |.
$$

If we write $U$ in matrix form $\begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$ in the logical basis, then the $4 \times 4$ matrix representation of the controlled $U$ gate is

$$
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & u_{00} & u_{01} \\
0 & 0 & u_{10} & u_{11}
\end{pmatrix}.
$$

All classical logic operations can be built up from the Hadamard gate, the CNOT gate, and a properly chosen phase gate.

Exercise 10.8 Show that the controlled $U$ gate is unitary.
10.2 Shor’s algorithm for factoring

Shor’s algorithm [28] for quantum computers breaks the security of RSA: it deduces a private key from a public key in polynomial time.

Reminder: The private key in RSA consists of two large primes \( p \) and \( q \), together with an exponent \( D \). The corresponding public key consists of the modulus \( N \),

\[
N = pq,
\]

and the public exponent \( E \) such that \( DE \equiv 1 \mod \varphi(N) \). Here \( \varphi \) stands for the Euler totient function, and \( \varphi(pq) = (p-1)(q-1) \). The security of RSA is based on the (empirically motivated) assumption that factoring large numbers is difficult. Hence deducing \( p \) and \( q \) from \( N \) is difficult. This implies that finding \( \varphi(N) \) is difficult too, which makes computing \( D \) from \( (N, E) \) difficult.

On a classical computer the best known factoring algorithm is the General Number Field Sieve [19]. It runs in sub-exponential time, of order \( \exp[n^{1/3}(\log n)^{2/3}] \). Here \( n \) is the size (in bits) of the public modulus \( N \). With the use of a quantum computer running Shor’s algorithm this can be reduced to \( O(n^3) \), a very significant speedup!

10.2.1 Preliminaries

The classical part of the Shor algorithm is based on the period of elements in \( \mathbb{Z}_N^* \). Remember, \( \mathbb{Z}_N^* \) is the multiplicative group modulo \( N \). Its elements are the integers \( \{ x : \gcd(x, N) = 1 \} \), i.e. the integers that do not have any factor in common with \( N \). The \( \varphi(N) \) precisely counts how many such elements there are. Even when the factorization of \( N \) is not known, the \( \gcd(x, N) \) is easily computed using the Euclidean algorithm.

The period of a group element \( a \in \mathbb{Z}_N^* \) is defined as the smallest integer \( r \) that satisfies

\[
a^r \equiv 1 \mod N. \tag{10.5}
\]

(Fermat’s little theorem \( a^{\varphi(N)} \equiv 1 \mod N \) tells us that the order of \( a \) has to be a divisor of \( \varphi(N) \).

In particular, \( r \leq \varphi(N) < N \).)

Suppose that we know the period of some \( a \). How much does that tell us about the factors of \( N \)? If \( r \) is even, (10.5) can be rewritten as

\[
(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \mod N. \tag{10.6}
\]

The factor \( (a^{r/2} - 1) \) cannot be a multiple of \( N \), since then \( r/2 \) would be a smaller integer than \( r \) with \( a^{-1} \equiv 1 \). It is well possible that the factor \( (a^{r/2} + 1) \) is a multiple of \( N \), but suppose that we find an \( a \) for which this is not the case. Then \( (a^{r/2} - 1) \) and \( (a^{r/2} + 1) \) each carry different factors of \( N \).

The classical part of Shor’s algorithm draws random elements of \( \mathbb{Z}_N^* \) and invokes a quantum computer to find the period; it does this until it has found some \( a \) with even period \( r \) such that \( (a^{r/2} + 1) \neq 0 \mod N \). Then the above reasoning is applied to find nontrivial factors of \( N \).

The quantum part of the algorithm is based on the Discrete Fourier Transform (DFT). First, a superposition is prepared of a large number of integer inputs \( x \). The function \( f(x) = a^x \mod N \) is applied to all \( x \) simultaneously, yielding a superposition of all the \( f(x) \) values. Then the DFT is applied to the ‘coordinate’ \( x \). The result is a superposition of all the Fourier-transformed coordinates \( k \), with amplitudes proportional to the Fourier coefficients \( \tilde{f}(k) \). A measurement of the state will with high probability return a value of \( k \) that has a large Fourier coefficient. From this \( k \) the period of \( a \) is finally extracted. The procedure is probabilistic, so it has to be repeated until a successful run occurs.
10.2.2 The classical part of the algorithm

1. Draw a random integer \( a < N \).
2. If \( \gcd(a, N) \neq 1 \) then you are amazingly lucky; you can find a nontrivial factor of \( N \).
3. Use the quantum subroutine to find the period of \( a \). (Denoted as \( r \).)
4. If \( r \) is odd go back to step 1.
5. If \( a^{r/2} + 1 \equiv 0 \mod N \) go back to step 1.
6. \( \gcd(a^{r/2} - 1, N) \) and \( \gcd(a^{r/2} + 1, N) \) are factors of \( N \).

10.2.3 The quantum part of the Shor algorithm

Determine \( Q = 2^q \) such that \( N^2 < Q < 2N^2 \). Define \( \omega = \exp i(2\pi/Q) \), so that \( \omega^Q = 1 \). There are two registers, each consisting of \( q \) qubits.

1. Prepare the initial state

\[
|\Psi_0\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle|0\rangle.
\]  

(10.7)

2. Apply the ‘quantum function’ \( f \), with \( f(x) := a^x \mod N \). This results in

\[
|\Psi_1\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle|f(x)\rangle.
\]  

(10.8)

3. Apply the discrete Fourier transform to the first register. The unitary operator is

\[
U_{\text{DFT}} = \frac{1}{\sqrt{Q}} \sum_{x,k=0}^{Q-1} \omega^{-kx} |k\rangle\langle x|.
\]

(10.9)

and the resulting state is

\[
|\Psi_2\rangle = U_{\text{DFT}}|\Psi_1\rangle = \frac{1}{Q} \sum_{k=0}^{Q-1} |k\rangle \otimes \sum_{x=0}^{Q-1} \omega^{-kx} |f(x)\rangle.
\]

(10.10)

We have \( f(x + r) = f(x) \), so lots of repetitions of \( f(x) \) occur in the \( x \)-sum. We can rewrite the \( x \)-summation for any function \( g \) as

\[
\sum_{x=0}^{Q-1} g(x) = \sum_{y=0}^{r-1} \sum_{j=0}^{j_{\text{max}}(y)} g(jr + y),
\]

(10.11)

with \( j_{\text{max}}(y) := \lfloor \frac{Q-1-y}{r} \rfloor \). Hence the state (10.10) is equal to

\[
|\Psi_2\rangle = \frac{1}{Q} \sum_{k=0}^{Q-1} \sum_{y=0}^{r-1} \omega^{-ky} \left[ \sum_{j=0}^{j_{\text{max}}(y)} \omega^{-jkr} \right] |k\rangle \otimes |f(y)\rangle.
\]

(10.12)

(Although \( r \) is unknown at this point, the equations still hold.)
4. Perform a measurement of both registers in the logical basis. The probability of ending up in a state \(|\tilde{\Psi}_2\rangle \otimes |f(y)\rangle\) is

\[
P_{ky} := \left| \langle \tilde{\Psi}_2 | (\tilde{\Psi} \otimes f(y)) \right|^2 = \frac{1}{\mathcal{Q}} \left| \sum_{j=0}^{j_{\text{max}}(y)} \omega^{jkr} \right|^2 = \frac{1}{\mathcal{Q}} \left| \sum_{j=0}^{j_{\text{max}}(y)} \omega^{-j(kr)\mathcal{Q}} \right|^2.
\]  

(10.13)

Here \(\{kr\}_{\mathcal{Q}}\) stands for the residue of \(kr\) modulo \(\mathcal{Q}\), but defined in such a way that the result lies in the interval \(-\frac{\mathcal{Q}}{2} < \{kr\}_{\mathcal{Q}} \leq \frac{\mathcal{Q}}{2}\). Notice that \(P_{ky}\) hardly depends on \(y\). The interesting point about this expression is that the probability vanishes unless \(kr\) is close to an integer multiple of \(\mathcal{Q}\). This can be seen as follows. Evaluation of the \(j\)-sum in (10.13) gives

\[
P_{ky} = \left| \frac{1}{\mathcal{Q}} \cdot \frac{1 - \omega^{-(kr)\mathcal{Q}[1+j_{\text{max}}(y)]}}{1 - \omega^{-(kr)\mathcal{Q}}} \right|^2 \approx \frac{1}{\mathcal{Q}} \cdot \frac{1 - \exp(-i2\pi \frac{kr}{\mathcal{Q}})}{1 - \exp(-i2\pi \frac{kr}{\mathcal{Q}})}.
\]  

(10.14)

Remember that \(\mathcal{Q} > N^2\), while \(r < N\). For the magnitude of \(\{kr\}_{\mathcal{Q}}\) we distinguish between three cases:

A: \(\{kr\}_{\mathcal{Q}}\) is of order \(\mathcal{Q}\);
B: \(r/2 \ll |\{kr\}_{\mathcal{Q}}| \ll \mathcal{Q}\);
C: \(\{kr\}_{\mathcal{Q}}\) is of order \(r/2\) or smaller.

For these three cases we examine the last expression in (10.14).

- Case A. The denominator in the fraction does not come close to zero. Consequently \(P_{ky}\) is of order \(1/\mathcal{Q}^2\) or smaller.
- Case B. The denominator gets close to zero. We can do a first order Taylor expansion \(\mathcal{Q}[1 - \exp(-i2\pi \frac{kr}{\mathcal{Q}})] \approx i2\pi \{kr\}_{\mathcal{Q}}\). Consequently \(P_{ky}\) is \(O(1/\{kr\}_{\mathcal{Q}}^2) \ll O(1/r^2)\).
- Case C. In subcase \(|\{kr\}_{\mathcal{Q}}| \ll r/2\), we can apply a first order Taylor expansion to both parts of the fraction, obtaining \(P_{ky} \approx 1/r^2\). Without rigorous proof, we mention the following result. If \(|\{kr\}_{\mathcal{Q}}| \leq r/2\) then \(P_{ky} \geq \frac{4}{\pi^2}r^2\).

Hence most of the probability is contained roughly in the interval \(\{|kr\}_{\mathcal{Q}}| \leq r/2\).

5. With high probability we now have a value \(k\) such that \(|kr - c\mathcal{Q}| \leq r/2\), for some unknown integer \(c\). Written differently, \(|\frac{k}{\mathcal{Q}} - \frac{c}{r}| \leq 1/(2\mathcal{Q})\). By writing a continued fraction of \(\frac{k}{\mathcal{Q}}\), an estimate \((c', r')\) is obtained (see Section 10.2.4). Look for candidates \((c', r')\) satisfying

\[
r' < N \quad \text{and} \quad \left| \frac{k}{\mathcal{Q}} - \frac{c'}{r'} \right| \leq \frac{1}{2\mathcal{Q}}.
\]  

(10.15)

If this estimate does not satisfy \(a^{r'} \equiv 1 \mod N\), then try neighbouring values of \(k\) or multiples of \(r'\).

6. If no solution was obtained in step 5, then go back to step 1.

**Exercise 10.9** Show how (10.14) follows from (10.13). Why is the approximation in (10.14) accurate?

**Exercise 10.10** Verify that the orders of magnitude for \(P_{ky}\) as listed below (10.14) are correct.

**Exercise 10.11** What would happen if the measurement of step 4 is repeated without going back to step 1?
10.2.4 Remarks about the Shor algorithm

The quantum Fourier transform

It is straightforward to verify that the operation (10.9) is unitary. We have

$$U_{DFT}^\dagger = \frac{1}{\sqrt{Q}} \sum_{x',k'=0}^{Q-1} \omega^{k'x'} |x'\rangle \langle k'|.$$

(10.16)

Multiplying from the left with $U_{DFT}$ and using the basis orthonormality $\langle x|x'\rangle = \delta_{x,x'}$ yields

$$U_{DFT} U_{DFT}^\dagger = \sum_{k,k'=0}^{Q-1} |k\rangle \langle k'| \frac{1}{Q} \sum_{x=0}^{Q-1} \omega^{x(k'-k)} = \sum_{k,k'=0}^{Q-1} |k\rangle \langle k'| \delta_{k,k'} = \sum_{k=0}^{Q-1} |k\rangle \langle k| = 1. \quad (10.17)$$

Performing the quantum DFT in practice is nontrivial. It has been shown that it can be done in theory with $O(n^2)$ quantum gates.

The modular exponentiation

This is the bottleneck of the quantum part of the algorithm. Running the function $f(x) = a^x \mod N$ as described in step 2 is the most time-consuming part of Shor’s algorithm.

Success probability

The probability of hitting a value of $k$ in step 4 that immediately leads to success in step 5 is quite large. The number of ‘good’ $c$ values (and hence $k$ values) is $\varphi(r)$. For every ‘good’ $k$, there are $r$ different states $|f(y)\rangle$ in $|\Psi_2\rangle$. Hence the total number of ‘good’ states is $r\varphi(r)$, and the total probability of hitting one of them is of order $r\varphi(r)/r^2 = \varphi(r)/r$. Numer theory tells us that $\varphi(r) = O(r/\log \log r)$ for large $r$. This gives $O(1/\log \log r)$ for the probability of immediate success. Even if the post-processing in step 5 was omitted (looking at neighbouring $k$ or multiples of $c$), the expected number of attempts would be only $O(\log \log r) = O(\log n)$.

Finding the approximate fraction (step 5)

Here we give an example of an algorithm that can be used to approximate $k/Q$ by a fraction $c'/r'$ with $r' < N$ and $\left| \frac{k}{Q} - \frac{c'}{r'} \right| \leq \frac{1}{2Q}$.

- $c_0 = 0; \quad r_0 = 1$
- $c_1 = 1; \quad r_1 = 0$
- $x = k/Q$
- for $i = 2, \ldots$
  - $a_i = \lfloor x \rfloor$
  - $c_i = a_ic_{i-1} + c_{i-2}$
  - $r_i = a_ir_{i-1} + r_{i-2}$
  - if $\left| \frac{k}{Q} - \frac{c_i}{r_i} \right| \leq \frac{1}{2Q}$ or $r_i \geq N$ then return
    - $x = 1/(x - a_i)$

The algorithm generates the continued fraction

$$\frac{c_i}{r_i} = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\ldots}}}. \quad (10.18)$$
10.3 Shor’s algorithm for discrete logarithms

[Section 10.3 is not part of the exam.]

10.3.1 Intuition

Let $p$ be a large prime. The discrete log problem is defined as follows. Given $p$, a generator $g$ of $\mathbb{Z}_p^*$ and $x \in \mathbb{Z}_p^*$, find an $r$ such that

$$x \equiv g^r \pmod{p}; \quad r \equiv \log_g(x) \pmod{p-1}. \quad (10.19)$$

We define $r$ to lie in the interval $-(p-3)/2 \leq r \leq (p-1)/2$, without loss of generality. The quantum algorithm for finding discrete logs is again based on period-finding. However, the function whose period needs to be found has two arguments, $f(u_1, u_2) := g^{u_1 x^{-u_2}} \pmod{p}$. Let $\delta \in \mathbb{Z}$. $f$ has the property

$$f(u_1 + r\delta, u_2 + \delta) = f(u_1, u_2), \quad (10.20)$$

i.e. the two-dimensional period is $\Delta \vec{u} = (r, 1)$. A two-dimensional DFT is applied to $f$, which results in strong probability peaks around wave vectors $\vec{k} \propto (1, -r)$, i.e. $\vec{k} \cdot \Delta \vec{u} = 0$.

10.3.2 The algorithm

The algorithm needs three registers. We define $Q$ as a power of 2, such that $p < Q < 2p$. The size of the registers (counted in qubits) is $\log_2(Q)$. The registers are prepared in the following state,

$$|\Psi_1\rangle = \frac{1}{p-1} \sum_{u_1, u_2=0}^{p-2} |u_1\rangle |u_2\rangle |g^{u_1 x^{-u_2}} \pmod{p}\rangle. \quad (10.21)$$

From now on, the first two registers will be denoted together, e.g. $|u_1\rangle |u_2\rangle = |\vec{u}\rangle$. A DFT is applied to the first and second register. Let $\omega = \exp(i 2\pi/Q)$. The DFT operator is

$$U_{DFT} = \frac{1}{Q} \sum_{k_1, k_2=0}^{Q-1} \sum_{u_1, u_2=0}^{p-2} \omega^{k_1 u_1} |k_1\rangle \langle k_1|$$

and the state after applying this operator is

$$|\Psi_2\rangle = U_{DFT} |\Psi_1\rangle = \frac{1}{Q(p-1)} \sum_{\vec{k}} \sum_{\vec{u}} \omega^{\vec{k} \cdot \vec{u}} |\vec{k}\rangle |g^{u_1 x^{-u_2}} \pmod{p}\rangle. \quad (10.22)$$

We write $g^{u_1 x^{-u_2}} \pmod{p} = g^{u_1 - ru_2} \pmod{p-1}$. For any $z \in \{0, \ldots, p-2\}$, the probability of measuring a state $|\vec{k}\rangle |g^z \pmod{p}\rangle$ is given by $|\langle \vec{k}, g^z \pmod{p} | \Psi_2 \rangle|^2$. The total probability of measuring $\vec{k}$ in the first two registers is

$$P_{\vec{k}} = \frac{1}{Q^2(p-1)^2} \sum_{z=0}^{p-2} \sum_{\vec{u}} \omega^{\vec{k} \cdot \vec{u}} \delta_{z, u_1 - ru_2} \pmod{p-1} \left| \langle \vec{k}, g^z \pmod{p} | \Psi_2 \rangle \right|^2. \quad (10.24)$$

We use a sum representation for the Kronecker delta. In general it holds for any positive $n$ that

$$\delta_{0, x \pmod{n}} = \frac{1}{n} \sum_{b=0}^{n-1} (e^{i 2\pi/n} b)^x. \quad (10.25)$$

We define $\alpha = e^{i \frac{2\pi}{p}}$. The Kronecker delta in (10.24) can be written as

$$\delta_{0, u_1 - ru_2 - z \pmod{p-1}} = \frac{1}{p-1} \sum_{h=0}^{p-2} \alpha^{h(u_1 - ru_2 - z)}. \quad (10.26)$$
substitution into (10.24) gives

\[ P_k = \frac{1}{Q^2(p-1)^2} \left| \sum_{z=0}^{p-2} \sum_{b=0}^{p-2} \alpha^{-zb} \sum_{u_1=0}^{Q-1} \omega^{k_1 u_1} \sum_{u_2=0}^{Q-1} \alpha^{-bru_2} \right|^2 \]

\[ = \frac{1}{Q^2(p-1)^2} \left| \sum_{z=0}^{p-2} \sum_{b=0}^{p-2} \alpha^{-zb} F_b \right|^2, \]

\[ F_b := \frac{1 - \alpha^{bQ}}{1 - \omega^{k_1} \alpha^b} \frac{1 - \alpha^{-brQ}}{1 - \omega^2 \alpha^{-br}}. \] (10.27)

the omegas in the numerator have disappeared because \( \omega^Q = 1 \). Note that \( F_b = 0 \) when \( bQ \) is an integer multiple of \( (p-1) \). It depends on the prime \( p \) how many powers of 2 are contained in \( p-1 \).

next we write the squared absolute value as \( |\sum_b \alpha^{-zb} F_b|^2 = \sum_{b,b'} \alpha^{z(b'-b)} F_b F_b^* \). The z-sum then yields a kronecker delta, \( \sum \alpha^{z(b'-b)} = (p-1) \delta_{b,b'} \). Together this gives us the following simplified result

\[ P_k = \frac{1}{Q^2(p-1)^2} \left| \sum_{b=0}^{p-2} |F_b|^2 \right. \] (10.28)

finally we study the behaviour of \( P_k \) as a function of \( \vec{k} \). let us write

\[ k_2 = -r k_1 + \delta \mod Q \] (10.29)

with \( |\delta| \leq Q/2 \). the \( \delta \) measures how far away the vector \( \vec{k} \) lies from the ‘preferred’ form \( (k_1, -r k_2) \) which matches the period of the function \( f(u_1, u_2) \). then

\[ F_b = \frac{1 - \alpha^{bQ}}{1 - \omega^{k_1} \alpha^b} \frac{1 - \alpha^{-brQ}}{1 - \omega^2 \alpha^{-br}}. \] (10.30)

we study the maxima of \( |F_b| \) as a function of \( b \). we have

\[ \omega^{k_1} \alpha^b = \exp i 2\pi \left( \frac{k_1}{Q} + \frac{b}{p-1} \right). \] (10.31)

we define \( b_*(k_1) \) as the value of \( b \) for which \( \omega^{k_1} \alpha^b \) is closest to 1.

\[ b_*(k_1) \equiv - \left\{ \frac{k_1(p-1)}{Q} \right\}_{\text{round}} = - \frac{k_1(p-1)}{Q} + \varepsilon(k_1), \] (10.32)

with \( |\varepsilon(k_1)| \leq \frac{1}{2} \). at \( b = b_*(k_1) \) we have

\[ \omega^{k_1} \alpha^{b_*(k_1)} = e^{i 2\pi \frac{\varepsilon(k_1)}{p-1}} \approx 1 + i 2\pi \frac{\varepsilon(k_1)}{p-1} \] (10.33)

which leads to a peak \( 1/(1 - \omega^{k_1} \alpha^{b_*(k_1)}) = \mathcal{O}(p/|\varepsilon(k_1)|) \).

when \( \varepsilon(k_1) = 0 \), the factors \( 1 - \alpha^{Q \delta} \) and \( 1 - \alpha^{r \delta Q} \) in \( F_b \) vanish; hence there is no peak at \( \varepsilon = 0 \). note however that for \( \varepsilon(k_1) \) nonzero but very close to 0, the factors \( 1 - \alpha^{b_*(k_1)Q} \) and \( 1 - \alpha^{-r \delta k_1 Q} \) in \( F_b \) generally do not vanish. hence most of the time when \( \varepsilon(k_1) \) comes close to 0, the \( F_b \) has a peak.

next we look at the behaviour of \( F_b \) for fixed \( k_1 \) and \( b = b_*(k_1) \), as a function of \( \delta \).

\[ \omega^\delta [\omega^{k_1} \alpha^{b_*(k_1)}]^{-r} = \omega^\delta \exp\left[-i \pi \varepsilon(k_1) \frac{2r}{p-1}\right] = \omega^\delta - \omega^\delta i \pi \varepsilon(k_1) \frac{2r}{p-1} + \mathcal{O}(|\varepsilon|/p)^2. \] (10.34)
Here we only consider $\varepsilon$ such that the $O(\varepsilon^2)$ terms are negligible. (Note that they are automatically negligible if $r$ is small.) For $\delta \approx 0$, the second fraction in (10.30) has a peak of order $\frac{p}{\pi(k_1)}$. Summarizing, at $b \approx b_\ast(k_1), \delta \approx 0$, the $F_b$ has a peak of order $\frac{p^2}{\pi^2(k_1)}$, which leads to a probability

$$P(k_1, -rk_1 \mod Q) = O\left(\frac{p}{Q^2 r^2 \varepsilon^4(k_1)}\right) = O\left(\frac{1}{pr^2 \varepsilon^4(k_1)}\right). \quad (10.35)$$

Success probability
The total probability contained in these peaks is obtained by summing over all $k_1$. Since $p$ is very large, such a sum is accurately approximated by an integral over $\varepsilon$. Using the fact that we have excluded $\varepsilon = 0$, and that for $\varepsilon \neq 0$ the $|\varepsilon|$ is uniformly distributed on $(\varepsilon_{\text{min}}, \cdots)$ with $\varepsilon_{\text{min}} = 1/Q$, we get

$$\text{Total prob. in peaks} = O\left(\int_{\varepsilon_{\text{min}}}^{\cdots} d\varepsilon \frac{1}{pr^2 \varepsilon^3}\right) = O\left(\frac{1}{pr^2 \varepsilon_{\text{min}}^3}\right) = O(1) \quad (10.36)$$

i.e. there is a considerable probability that a measurement yields $\tilde{k}$ such that $k_2 \approx -rk_1 \mod Q$. This probability does not vanish with increasing $p$. 
Bibliography


