An Introduction to Cryptography

Mohamed Barakat, Christian Eder, Timo Hanke

September 20, 2018

Preface

Second Edition

Lecture notes of a class given during the summer term 2017 at the University of Kaiserslautern. The notes are based on lecture notes by Mohamed Barakat and Timo Hanke [BH12] (see also below). Other good sources and books are, for example, [Buc04, Sch95, MVO96].

Many thanks to Raul Epure for proofreading and suggestions to improve the lecture notes.

First Edition

These lecture notes are based on the course “Kryptographie” given by Timo Hanke at RWTH Aachen University in the summer semester of 2010. They were amended and extended by several topics, as well as translated into English, by Mohamed Barakat for his course “Cryptography” at TU Kaiserslautern in the winter semester of 2010/11. Besides the literature given in the bibliography section, our sources include lectures notes of courses held by Michael Cuntz, Florian Heß, Gerhard Hiß and Jürgen Müller. We would like to thank them all.

Mohamed Barakat would also like to thank the audience of the course for their helpful remarks and questions. Special thanks to Henning Kopp for his numerous improvements suggestions. Also thanks to Jochen Kall who helped locating further errors and typos. Daniel Berger helped me with subtle formatting issues. Many thanks Daniel.

Contents

Chapter 1

Introduction

Cryptology consists of two branches:

Cryptography is the area of constructing cryptographic systems.

Cryptanalysis is the area of breaking cryptographic systems.

Cryptography is a field of computer science and mathematics that focusses on techniques for secure communication between two parties (Alice & Bob) while a third-party (Eve[1] or Mallory[2]) is present (see Figure 1.1). This is based on methods like encryption, decryption, signing, generating of pseudo random numbers, etc.

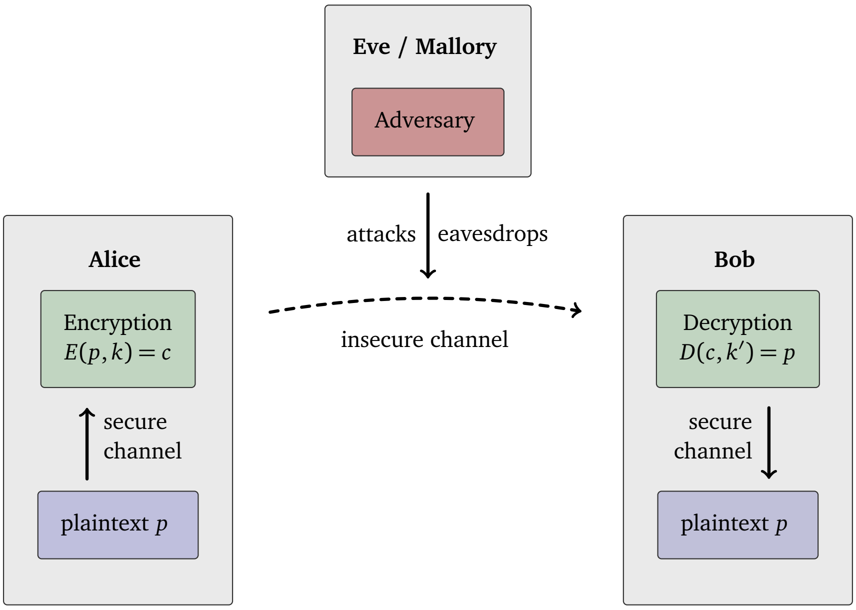


Figure 1.1: A basic idea for secure communication

[1] Usually "Eve" stands for eavesdropper.

[2] "Mallory" stands for a man-in-the-middle attack.

The four ground principles of cryptography are

Confidentiality Defines a set of rules that limits access or adds restriction on certain information.

Data Integrity Takes care of the consistency and accuracy of data during its entire life-cycle.

Authentication Confirms the truth of an attribute of a datum that is claimed to be true by some entity.

Non-Repudiation Ensures the inability of an author of a statement resp. a piece of information to deny it.

Nowadays there are in general two different schemes: On the one hand, there are *symmetric schemes*, where both, Alice and Bob, need to have the same key in order to encrypt their communication. For this, they have to securely exchange the key initially. On the other hand, since Diffie and Hellman's key exchange idea from 1976 (see also Example 1.1 (3) and Chapter 8) there also exists the concept of *asymmetric schemes* where Alice and Bob both have a private and a public key. The public key can be shared with anyone, so Bob can use it to encrypt a message for Alice. But only Alice, with the corresponding private key, can decrypt the encrypted message from Bob.

In this lecture we will discover several well-known cryptographic structures like RSA (Rivest-Shamir-Adleman cryptosystem), DES (Data Encryption Standard), AES (Advanced Encryption Standard), ECC (Elliptic Curve Cryptography), and many more. All these structures have two main aspects:

1. There is the security of the structure itself, based on mathematics. There is a standardization process for cryptosystems based on theoretical research in mathematics and complexity theory. Here our focus will lay in this lecture.
2. Then we have the implementation of the structures in devices, e.g. SSL, TLS in your web browser or GPG for signed resp. encrypted emails. These implementations should not diverge from the theoretical standards, but must still be very fast and convenient for the user.

It is often this mismatch between these requirements that leads to practical attacks of theoretically secure systems, e.g. [Wik16b, Wik16c, Wik16e].

Before we start defining the basic notation let us motivate the following with some historically known cryptosystems:

Example 1.1.

1. One of the most famous cryptosystems goes back to Julius Ceasar: *Caesar's cipher* does the following: Take the latin alphabet and apply a mapping $A \mapsto 0, B \mapsto 1, \ldots, Z \mapsto 25$. Now we apply a shifting map
$$x \mapsto (x + k) \mod 26$$
for some secret $k \in \mathbb{Z}$. For example, `ATTACK` maps to `CVVCEM` for $k = 2$. This describes the encryption process. The decryption is applied via the map
$$y \mapsto (y - k) \mod 26$$
with the same k. Clearly, both parties need to know k in advance. Problems with this cipher: Same letters are mapped to the same shifted letters, each language has its typical distribution of letters, e.g. `E` is used much more frequently in the English language than `K`. Besides investigating only single letters one can also check for letter combinations of length 2-3, etc.

2. A generalization of Caesar's cipher is *Vigenère's cipher*: It was invented several times, nowadays the reference goes back to the French cryptographer Blaise de Vigenère. The main difference is that instead of using only one $k \in \mathbb{Z}$, we now use $k \in \mathbb{Z}^n$ for some $n \in \mathbb{N}$. For example, let the secret be represented by the word SECRET. We again map the letters from the alphabet to corresponding numbers modulo 26:

   $$k = (18, 4, 2, 17, 4, 19) \in (\mathbb{Z}/26\mathbb{Z})^6 .$$

   Now we apply for each letter the Caesar cipher to our text ATTACK:

   $$\begin{array}{ccc} \texttt{A} & \mapsto & \texttt{S} \\ \texttt{T} & \mapsto & \texttt{X} \\ \texttt{T} & \mapsto & \texttt{V} \\ \texttt{A} & \mapsto & \texttt{R} \\ \texttt{C} & \mapsto & \texttt{G} \\ \texttt{K} & \mapsto & \texttt{C} \end{array}$$

   This system is a bit harder to attack, try to find redundancies in the text like the letter E appearing on several positions. With this one can crack the length of the secret key n. Afterwards one can splice the text in chunks of n letters and rather easily test all possibilities via some assumptions like the text contains English words from the dictionary etc. Still note that k has to be known to Alice and Bob at the same time.

3. In 1976 Whitfield Diffie and Martin Hellman (and also Ralph Merkle) proposed an idea for securely exchanging keys over an insecure communication channel, nowadays known as *Diffie-Hellman Key Exchange*:

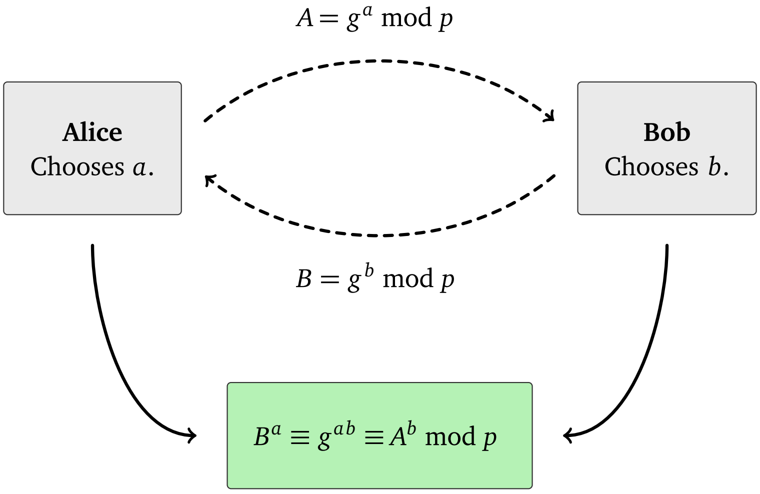


Figure 1.2: Diffie-Hellman Key Exchange

   a) Alice and Bob agree publicly on a cyclic group, e.g. $G = \langle g \rangle$, $G = \mathbb{F}_p^*$.

b) Alice chooses randomly some $0 \leq a < |G|$ and computes $A := g^a$. Bob chooses randomly some $0 \leq b < |G|$ and computes $B := g^b$.

c) Alice sends Bob A. Bob sends Alice B.

d) Alice computes $S := B^a = (g^b)^a = g^{ab}$. Bob computes $S := B^a = (g^a)^b = g^{ab}$.

e) Now Alice and Bob can use S as their secret key to encrypt and decrypt messages.

Outside of this process Eve only knows $G = \langle g \rangle$, A and B, but she does not know a, b, S. Thus Eve either needs to compute $a = \log_g A$ and $b = \log_g B$ (this is known as *discrete logarithm problem* and is assumed to be "hard"); or she has some other magical function f such that $S = f(A, B, G)$. Clearly, security of this system highly relies on the choice of the group, i.e. g. For example, taking $G = (\mathbb{Z}/n\mathbb{Z}, +) = \langle 1 \rangle$, thus exponentiation g^a boils down to $g \cdot a = a$ in this setting.

Note that there might also be Mallory, a *(wo)man-in-the-middle*: Mallory might tell Alice to be Bob and does a Diffie-Hellman key exchange with Alice getting a secret S. In the same way, Mallory tells Bob to be Alice and they are doing another key exchange getting S'. Whenever Alice sends Bob a message, Mallory takes the encrypted message, decrypts it with S, reads it, and encrypts it with S'. Then the newly encrypted message is sent to Bob. If Mallory can get hold of any message Alice and Bob sends each other, Alice and Bob will not be able to realize this attack.

□

Chapter 2

Basic Concepts

We define basic notations and formal definitions for the main structures we are working on in the following.

2.1 Quick & Dirty Introduction to Complexity Theory

Definition 2.1. An algorithm[1] is called **deterministic** if the output only depends on the input. Otherwise we call it **probabilistic** or **randomized**. □

Definition 2.2. Let $f, g : \mathbb{N} \to \mathbb{R}$ be two functions. We denote $f(n) = \mathcal{O}(g(n))$ for $n \to \infty$ iff there is a constant $M \in \mathbb{R}_{>0}$ and an $N \in \mathbb{N}$ such that $|f(n)| \leq M|g(n)|$ for all $n \geq N$. In general $\mathcal{O}(g)$ denotes the set

$$\mathcal{O}(g) = \{h : \mathbb{N} \to \mathbb{R} \mid \exists M_h \in \mathbb{R}_{>0} \exists N \in \mathbb{N} : |h(n)| \leq M_h |g(n)| \forall n \geq N\}.$$

We are always interested in the growth rate of the function for $n \to \infty$, so usually we write $f = \mathcal{O}(g)$ (equivalend to $f \in \mathcal{O}(g)$) as a shorthand notation. □

Example 2.3. Let $f, g : \mathbb{N} \to \mathbb{R}$ be two functions.

1. $f = \mathcal{O}(1)$ iff $f : \mathbb{N} \to \mathbb{R}$ and f is bounded.
2. $\mathcal{O}\left(3n^2 + 1782n - 2\right) = \mathcal{O}\left(-17n^2\right) = \mathcal{O}\left(n^2\right)$.
3. If $|f| \leq |g|$, i.e. $|f(n)| \leq |g(n)|$ for all $n \in \mathbb{N}$, then $\mathcal{O}(f) \subset \mathcal{O}(g)$.

□

Lemma 2.4. Let $f, g : \mathbb{N} \to \mathbb{R}$ be two functions.

1. $f = \mathcal{O}(f)$.
2. $c\mathcal{O}(f) = \mathcal{O}(f)$ for all $c \in \mathbb{R}_{\geq 0}$.
3. $\mathcal{O}(f)\mathcal{O}(g) = \mathcal{O}(fg)$.

□

[1] No, we do not start discussing what an algorithm is.

Proof. Exercise. ■

Definition 2.5.

1. Let $x \in \mathbb{Z}_{\geq 0}$ and $b \in \mathbb{N}_{>1}$. Then we define the **size of** x **w.r.t.** b by $\operatorname{sz}_b(x) := \lfloor \log_b(x) \rfloor + 1 \in \mathbb{N}$.[2] There exist then $(x_1, \ldots, x_{\operatorname{sz}_b(x)}) \in \{0, \ldots, b-1\}^{\operatorname{sz}_b(x)}$ such that

   $$x = \sum_{i=1}^{\operatorname{sz}_b(x)} x_i b^{\operatorname{sz}_b(x)-i}$$

   is the b**-ary repesentation** of x. For a $y \in \mathbb{Z}$ we define $\operatorname{sz}_b(y) := \operatorname{sz}_b(|y|) + 1$ where the additional value encodes the sign.

2. The **runtime** of an algorithm for an input x is the number of elementary steps of the algorithm when executed by a multitape Turing machine.[3] The algorithm is said to **lie in** $\mathcal{O}(f)$ if the runtime of the algorithm is bounded (from above) by $f(\operatorname{sz}_b(x))$.

3. An algorithm is called a **polynomial (runtime) algorithm** if it lies in $\mathcal{O}\left(n^k\right)$ for some $k \in \mathbb{N}$ and input of size n. Otherwise it is called an **exponential (runtime) algorithm**.

□

Example 2.6.

1. The 2-ary, i.e. binary, representation of 18 is $(1,0,0,1,0) \in (\mathbb{Z}_2)^5$ resp. $18 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$. Its size is $\operatorname{sz}_2(18) = 5$.

2. Addition of two bits, i.e. two number of binary size 1 lies in $\mathcal{O}(1)$. Addition and subtraction of two natural numbers a, b of sizes m resp. n in schoolbook method lies in $\mathcal{O}(\max\{m, n\})$.

3. Multiplication of two natural numbers of binary size n lies in $\mathcal{O}\left(n^2\right)$ with the schoolbook method. It can be improved, for example, by the Schönhage-Strassen multiplication algorithm that lies in $\mathcal{O}(n \log n \log \log n)$.

□

With these definitions we can classify the complexity of problems.

Definition 2.7. A problem instance P lies in the complexity class

1. P if P is solvable by a deterministic algorithm with polynomial runtime.
2. BPP if P is solvable by a probabilistic algorithm with polynomial runtime.
3. BQP if P is solvable by a deterministic algorithm on a quantum computer in polynomial runtime.
4. NP if P is verifiable by a deterministic algorithm with polynomial runtime.
5. NPC if any other problem in NP can be reduced resp. transformed to P in polynomial time.
6. EXP if P is solvable by a deterministic algorithm with exponential runtime.

[2] For $y \in \mathbb{Z}$ the floor function is defined by $\lfloor y \rfloor = \max\{k \in \mathbb{Z} \mid k \leq y\}$.

[3] Think of a computer.

□

Remark 2.8.

1. It is known that:

   a) $\mathsf{P} \subset \mathsf{BPP}$ and $\mathsf{NP} \subset \mathsf{EXP}$. Still, the relation between BPP and NP is unknown.

   b) $\mathsf{P} \subsetneq \mathsf{EXP}$, whereas for the other inclusions strictness is not clear.

   c) Factorization of a natural number and the discrete logarithm problem lie in $\mathsf{NP} \cap \mathsf{BQP}$.

   It is conjectured that:

   a) $\mathsf{P} = \mathsf{BPP}$.

   b) Factorization of a natural number and the discrete logarithm problem do not lie in BPP.

   c) $\mathsf{NP} \neq \mathsf{BQP}$, in particular $\mathsf{NPC} \cap \mathsf{BQP} = \emptyset$.

So the wet dream of any cryptographer is that there exists a problem $P \in \mathsf{NP} \setminus \mathsf{BQP}$. □

Definition 2.9. We call an algorithm resp. a problem **feasible** if it lies in P, BPP or BQP. Otherwise the algorithm is **suspected** resp. **expected to be infeasible**.[4] □

2.2 Underlying Structures

First we define a special kind of mapping:

Definition 2.10. Let M, N be sets. A **multivalued map** from M to N is a map $F : M \to 2^N$ with[5] $F(m) \neq \emptyset$ for all $m \in M$. We use the notation $F : M \rightsquigarrow N$ and write $F(m) = n$ for $n \in F(m)$. Moreover, we define the following properties:

1. F is **injective** if the sets $F(m)$ are pairwise disjoint for all $m \in M$.
2. F is **surjective** if $\cup_{m \in M} F(m) = N$.
3. F is **bijective** if it is injective and surjective.
4. Let $F : M \rightsquigarrow N$ be surjective, then we define the **multivalued inverse** F^{-1} of F via

$$F^{-1} : N \rightsquigarrow M, F^{-1}(n) := \{m \in M \mid F(m) = n\} = \{m \in M \mid n \in F(m)\}.$$

5. Let $F : M \rightsquigarrow N$ and $F' : M \rightsquigarrow N$ be two multivalued maps. We write $F \subset F'$ if $F(m) \subset F'(m)$ for all $m \in M$.

□

Lemma 2.11. Let $F : M \rightsquigarrow N$ be a multivalued map. F defines a map $M \to N$ iff $|F(m)| = 1$ for all $m \in M$. □

Proof. Clear by Definition 2.10. ■

[4] Note that this is not a completely accurate definition of the terms feasible and infeasible. We refer to [Wik17b] for more information on this topic.

[5] 2^N denotes the power set of N.

Remark 2.12. If a multivalued map $F : M \rightsquigarrow N$ defines a map $M \to N$ then we say that F **is** this map and denote the corresponding map also by F. □

Example 2.13.

1. Let $M = \{1, 2, 3\}$ and $N = \{A, B, C, D\}$. Then we can define several multivalue maps $F : M \rightsquigarrow N$, for example:

   a) $1 \mapsto \{A\}, 2 \mapsto \{B\}, 3 \mapsto \{C, D\}$. Then F is surjective and injective, thus bijective.

   b) $1 \mapsto \{B\}, 2 \mapsto \{A, C\}, 3 \mapsto \{B, D\}$. Then F is surjective, but not injective as $F(1) \cap F(3) = \{B\} \neq \emptyset$.

   c) $1 \mapsto \{A\}, 2 \mapsto \{C\}, 3 \mapsto \{D\}$. Then F is injective, but not surjective. It holds that $|F(m)| = 1$ for all $m \in M$ thus F is a map namely the injective map $F : M \to N$.

2. For $a \in \mathbb{R}_{>0}$ the equation $x^2 - a = 0$ has two solutions. We can define a corresponding multivalued map $F : \mathbb{R}_{>0} \rightsquigarrow \mathbb{R}$ via mapping $a \in \mathbb{R}_{>0}$ to $\{\pm\sqrt{a}\} \subset \mathbb{R}$.

3. Take any surjective (not necessarily injective) function $f : N \to M$ between sets M, N. One can construct a corresponding multivalued map $F : M \rightsquigarrow N$ by taking the inverse relation (note that the inverse **function** need not exist) of f.

□

Definition 2.14. An **alphabet** is a non-empty set Σ. We denote the **length** resp. **size** resp. **cardinality of** Σ by $|\Sigma| = \#\Sigma$. Elements of Σ are called **letters** or **digits** or **symbols**.

1. A **word** over Σ is a finite sequence of letters of Σ. The empty sequence denotes the empty word ε. The **length of a word** w is denoted by $|w| \in \mathbb{N}$; by definition, $|\varepsilon| = 0$. Moreover, let Σ^n denote the set of all words of length n for any $n \in \mathbb{N}$. Then we can write any $w \in \Sigma^n$ as $w = (w_1, \ldots, w_n)$.

2. We define the set of "all words" resp. "all texts" by[6]

$$\Sigma^\bullet := \cup_{i=0}^{\infty} \Sigma^i.$$

3. On $\Sigma^\bullet$ we have the **concatenation** $\circ$ of two words as binary operation: Let $v \in \Sigma^m$ and $w \in \Sigma^n$ then

$$vw := v \circ w = (v_1, \ldots, v_m) \circ (w_1, \ldots, w_n) = (v_1, \ldots, v_m, w_1, \ldots, w_n) \in \Sigma^{m+n}$$

   such that $|vw| = |v| + |w|$. In particular: $v \circ \varepsilon = \varepsilon \circ v = v$.

4. A **formal language** L is a subset $L \subset \Sigma^\bullet$.

□

Remark 2.15. We note that $(\Sigma^\bullet, \circ)$ is a semi-group (closed and associative) with neutral element ε. Moreover, $|*| : (\Sigma^\bullet, \circ) \to (\mathbb{Z}_{\geq 0}, +)$ is a semi-group homomorphism. Clearly, in general $\Sigma^\bullet$ is not commutative. In particular, it holds: $\Sigma^\bullet$ is commutative iff $|\Sigma| = 1$. □

Example 2.16.

[6] In formal language theory $\Sigma^\bullet$ is also called Kleene star or Kleene closure.

1. In English language we could use an alphabet

   $$\Sigma = \{A,\dots,Z,a,\dots,z,0,\dots,9,\ \ ,,,.,:,;,@\}.$$

   Now, for example, any email address containing the above symbols is a word in $\Sigma^\bullet$. Note that we cannot write any English text, for example, “-” or “!” are not included in Σ.

2. On a computer we can use $\Sigma = \mathbb{F}_2 = \{0,1\}$ and some encodings like `ASCII`: 7-bits are encoded in characters, so we have all $2^7 = 128$ possible words from Σ^7: The first 32 are reserved for control characters (non-printable), then the printable characters start. For example:

binary, i.e. $\Sigma^7 = \mathbb{F}_2^7$	decimal	glyph
010 0001	33	`!`
011 0111	55	`7`
101 0111	87	`W`
111 1010	122	`z`

□

Definition 2.17. A **cryptosystem** is a 5-tuple $\Pi := (\mathscr{P},\mathscr{C},\kappa,\mathscr{E},\mathscr{D})$ where

1. $\mathscr{P} \subset \Sigma_1^\bullet$, $\mathscr{C} \subset \Sigma_2^\bullet$ for alphabets Σ_1,Σ_2,
2. $\kappa : \mathscr{K}' \to \mathscr{K}$ is a bijective map between sets $\mathscr{K},\mathscr{K}'$,
3. $\mathscr{E} = (\mathscr{E}_e)_{e\in\mathscr{K}}$ is a family of multivalued maps $\mathscr{E}_e : \mathscr{P} \rightsquigarrow \mathscr{C}$, and
4. $\mathscr{D} = (\mathscr{D}_d)_{d\in\mathscr{K}'}$ is a family of surjective maps $\mathscr{D}_d : \mathscr{C} \mapsto \mathscr{P}$,

such that

$$\mathscr{E}_{\kappa(d)} \subset \mathscr{D}_d^{-1} \text{ for all } d \in \mathscr{K}' \text{ interpreted as multivalued maps.} \tag{2.1}$$

We further require that $\mathscr{E}$ and $\mathscr{D}$ are realized by polynomial runtime algorithms where $\mathscr{E}$ may be probabilistic. Moreover, we call

1. Σ_1 the **plaintext alphabet** and $\mathscr{P}$ the **set of plaintexts**,
2. Σ_2 the **ciphertext alphabet** and $\mathscr{C}$ the **set of ciphertexts**,
3. $\mathscr{K}$ resp. $\mathscr{K}'$ the **encryption** resp **decryption key space**, their elements are called **keys** and κ is called the **key correspondence**,
4. $\mathscr{E}$ the **encryption algorithm** resp. $\mathscr{E}_e$ the **encryption algorithm with key** e, and
5. $\mathscr{D}$ the **decryption algorithm** resp. $\mathscr{D}_d$ the **decryption algorithm with key** d.

□

Remark 2.18. Let us try to clarify the meaning of different parts of Definition 2.17:

1. Often $\mathscr{K} = \mathscr{K}'$ and κ is just the identity map.
2. Likewise, often $\Sigma_1 = \Sigma_2 = \Sigma$ and $\mathscr{P} = \Sigma^\bullet$.

3. $\mathscr{E}_e$ being a multivalued map is no problem as long as Equation 2.1 holds. It ensures that even if the encryption leads to a set of ciphertexts this set is in the preimage of corresponding decryption algorithm. Moreover, it follows that the multivalued map $\mathscr{E}_e$ is injective for all $e \in \mathscr{K}$.

4. In particular, we often assume that both, $\mathscr{E}$ and $\mathscr{D}$, are families of maps, in particular, $\mathscr{E}$ is a family of injective maps $\mathscr{P} \to \mathscr{C}$. Moreover, it then holds by construction that

$$\begin{array}{rrcl} \forall d \in \mathscr{K}' & \mathscr{E}_{\kappa(d)} \circ \mathscr{D}_d & = & \mathrm{id}_{\mathscr{C}}, \\ \forall e \in \mathscr{K} & \mathscr{D}_{\kappa^{-1}(e)} \circ \mathscr{E}_e & = & \mathrm{id}_{\mathscr{P}}. \end{array}$$

□

Example 2.19. Recall Example 1.1:

1. We can describe Caesar's cryptosystem via

$$\Sigma_1 = \Sigma_2 = \Sigma = \mathscr{P} = \mathscr{C} = \mathscr{K} = \mathscr{K}' = \{A, \ldots, Z\} \cong \mathbb{Z}/26\mathbb{Z}, \text{ and}$$

$$\begin{array}{rcccccl} \mathscr{E}_k : & \mathscr{P} & \to & \mathscr{C}, & m & \mapsto & (m+k) \bmod 26, \\ \mathscr{D}_\ell : & \mathscr{C} & \to & \mathscr{P}, & c & \mapsto & (c-\ell) \bmod 26. \end{array}$$

In other words: $\mathscr{E}_k = \mathscr{D}_{-k}$ and thus $\mathscr{E}_k$ is a map for all $k \in \mathscr{K}$.

2. The Vigenère cryptosystem now generalizes the Caesar one to: We still have $\Sigma_1 = \Sigma_2 = \Sigma \cong \mathbb{Z}/26\mathbb{Z}$ and $\mathscr{K} = \mathscr{K}'$, but now $\mathscr{P} = \mathscr{C} = \mathscr{K} = \Sigma^{|k|}$ for a key k. In our example we have

$$k = \text{“}\texttt{SECRET}\text{”} = (18, 4, 2, 17, 4, 19) \in (\mathbb{Z}/26\mathbb{Z})^6.$$

Thus we can assume $\mathscr{P}, \mathscr{C} \subset \Sigma^6$ (if the texts are longer we can cut them in blocks of length 6) and we can apply $\mathscr{E}$ and $\mathscr{D}$ component wise:

$$\begin{array}{rcccccccl} \mathscr{E}_k : & \mathscr{P} & \to & \mathscr{C}, & m & \mapsto & (m+k) \bmod 26 & = & (m_1+k_1, \ldots, m_6+k_6) \bmod 26, \\ \mathscr{D}_\ell : & \mathscr{C} & \to & \mathscr{P}, & c & \mapsto & (c-\ell) \bmod 26 & = & (c_1-\ell_1, \ldots, c_6-\ell_6) \bmod 26. \end{array}$$

Again it holds that $\mathscr{E}_k = \mathscr{D}_{-k}$ and thus $\mathscr{E}_k$ is a map for all $k \in \mathscr{K}$.

□

Definition 2.20. A **cipher**[7] is an algorithm for performing encryption or decryption. If we have a cryptosystem, the corresponding cipher is given by $\mathscr{E}$ resp. $\mathscr{D}$ (implicitly also the keyspaces $\mathscr{K}$ or $\mathscr{K}'$ resp. κ). In the following we use the terms **cryptosystem** and **cipher** synonymously to each other.

There are two main categories of ciphers in terms of key handling: If κ is feasible then $\mathscr{K}$ and $\mathscr{K}'$ need to be kept secret and the cipher is called **symmetric**. Otherwise the cipher is called **asymmetric**. We also call a cryptosystem **symmetric** resp. **asymmetric** if its corresponding cipher is symmetric resp. asymmetric. An asymmetric cryptosystem is also called a **public key cryptosystem** as $\mathscr{K}$ can be made public without weakening the secrecy of the “private” key set $\mathscr{K}'$ for decryption. The elements of $\mathscr{K}$ are then called **public keys**, those of $\mathscr{K}'$ are called **private keys**.
□

[7] Some authors also write **cypher**, in German it stands for **Chiffre**.

Remark 2.21. Implementations of symmetric cryptosystems are more efficient than those of asymmetric cryptosystems. Thus, asymmetric ciphers are in general only used for exchanging the needed private keys in order to start a secure communication via a symmetric cipher. □

Example 2.22. Both, Ceasar's and Vigenère's cryptosystem, are symmetric ones as encryption by the key k is decrypted with the key $-k$. □

In the early days of cryptography the systems resp. ciphers were kept secret.[8] Doing so is no longer possible nowadays and also has the disadvantage that no research on the security of a cryptosystem kept secret can be done. Thus, the following principle is widely accepted in cryptology:

Principle 2.23 (Kerckhoff, 1883)**.** The cryptographic strength of a cryptosystem should not depend on the secrecy of the cryptosystem, but only on the secrecy of the decryption key. □

In other words: The attacker always knows the cryptosystem.

2.3 Investigating Security Models

Definition 2.24. For a given cryptosystem Π we define the following **security properties**:

1. Π has **onewayness** (OW) if it is infeasbile for an attacker to decrypt an arbitrary ciphertext.
2. Π has **indistinguishability** (IND) if it is infeasible for an attacker to associate a given ciphertext to one of several known plaintexts.
3. Π has **non-malleability** (NM) if it is infeasible for an attacker to modify a given ciphertext in a way such that the corresponding plain text is sensible in the given language resp. context.

□

Remark 2.25. It is known that $\mathsf{NM} \Rightarrow \mathsf{IND} \Rightarrow \mathsf{OW}$. □

Definition 2.26.

1. An **active attack** on a cryptosystem is one in which the attacker actively changes the communication by, for example, creating, altering, replacing or blocking messages.
2. A **passive attack** on a cryptosystem is one in which the attacker only eavesdrops plaintexts and ciphertexts. In contrast to an active attack the attacker cannot alter any messages she/he sees.

□

In this lecture we are mostly interested in passive attacks. Some attack scenarios we might consider are presented in the following:

Definition 2.27.

1. The attacker receives only ciphertexts: **ciphertext-only attack** (COA).

[8] Security by obscurity.

2. The attacker receives pairs of plaintexts and corresponding ciphertexts: **known-plaintext attack** (KPA).
3. The attacker can for one time choose a plaintext and receives the corresponding ciphertext. He cannot alter his choice depending on what he receives: **chosen-plaintext attack** (CPA).
4. The attacker is able to adaptively choose ciphertexts and to receive the corresponding plaintexts. The attacker is allowed to alter the choice depending on what is received. So the attacker has access to the decryption cipher $\mathscr{D}_d$ and wants to get to know the decryption key d: **adaptive chosen-ciphertext attack** (CCA).

□

Remark 2.28. In a public cryptosystem CPA is trivial. Moreover, one can show that in general it holds that

$$\mathsf{CCA} > \mathsf{CPA} > \mathsf{KPA} > \mathsf{COA}.$$

□

Definition 2.29. A **security model** is a security property together with an attack scenario. □

Example 2.30. IND-CCA is a security model. One would check the indistinguishability of a given cryptosystem Π w.r.t. an adaptive chosen-ciphertext attack. □

Chapter 3

Modes of Ciphers

For ciphers we have, in general, four different categories:

1. symmetric and asymmetric ciphers (see Definition 2.20), and
2. stream and block ciphers.

In the following we often assume binary representation of symbols, i.e. we are working with bits in $\mathbb{Z}/2\mathbb{Z}$. All of what we are doing can be easily generalized to other representations and other alphabets.

3.1 Block Ciphers

Definition 3.1. Let Σ be an alphabet. A **block cipher** is a cipher acting on $\mathscr{P} = \mathscr{C} = \Sigma^n$ for a given **block size** $n \in \mathbb{N}$. Block ciphers with block size $n = 1$ are called **substitution ciphers**. □

Lemma 3.2. The encryption functions of block ciphers are the permutations on Σ^n. □

Proof. By definition the encryption functions $\mathscr{E}_e$ are injective for each $e \in \mathscr{K}$. Injective functions $\mathscr{E}_e : \Sigma^n \to \Sigma^n$ are bijective, thus permutations on Σ^n. ■

If we assume $\mathscr{P} = \mathscr{C} = \Sigma^n$ the keyspace $\mathscr{K}' = \mathscr{K} = S(\Sigma^n)$ is the set of all permutations on Σ^n. Depending on Σ and n, $S(\Sigma^n)$ is huge, having $(|\Sigma|^n)!$ elements. In practice one chooses only a subset of $S(\Sigma^n)$ such that the permutations can be generated easily by short keys. Clearly, this might install a security problem to the cryptosystem.

A special case of this restriction is to use the permutation group S_n on the positions as key space:

Example 3.3. A **permutation cipher** is a block cipher that works on $\mathscr{P} = \mathscr{C} = \Sigma^n$ for some $n \in \mathbb{N}$ and uses $\mathscr{K}' = \mathscr{K} = S_n$. In this way $|\mathscr{K}'| = n!$ which is much smaller than $|S(\Sigma^n)|$. Let $\pi \in \mathscr{K}$:

$$\begin{aligned} \mathscr{E}_\pi : \quad & \Sigma^n \to \Sigma^n, \quad (v_1, \ldots, v_n) \mapsto \left(v_{\pi(1)}, \ldots, v_{\pi(n)}\right), \\ \mathscr{D}_{\pi^{-1}} : \quad & \Sigma^n \to \Sigma^n, \quad (v_1, \ldots, v_n) \mapsto \left(v_{\pi^{-1}(1)}, \ldots, v_{\pi^{-1}(n)}\right). \end{aligned}$$

For example, let $n = 3$ and $\Sigma = \mathbb{Z}/2\mathbb{Z}$. We use the keyspaces

$$\mathscr{K}' = \mathscr{K} = S_3 = \{(1), (1\,2), (1\,3), (2\,3), (1\,2\,3), (1\,3\,2)\}$$

with 3! elements. Now we could, for example, encrypt the plaintext $(v_1, v_2, v_3) = (1,0,1) \in \Sigma^3$ via $\pi = (1\,2\,3)$: $\mathscr{E}_\pi((1,0,1)) = (v_2, v_3, v_1) = (0,1,1)$. □

For a symmetric block cipher one can increase the level of security to multiple applications of the the cipher:

Remark 3.4. Let $\mathscr{E}$ and $\mathscr{D}$ represent a symmetric block cipher of block size n, w.l.o.g. we assume $\mathscr{K}' = \mathscr{K}$. An easy way to increase its security is to apply the so-called **triple encryption**: Take three keys $k_1, k_2, k_3 \in \mathscr{K}$. Then one can encrypt a plaintext $p \in \mathscr{P}$ via

$$\mathscr{E}_{k_3}\left(\mathscr{D}_{k_2}\left(\mathscr{E}_{k_1}(p)\right)\right) =: c \in \mathscr{C}.$$

There are three different settings for the keys:

1. If $k_1 = k_2 = k_3$ the above encryption is equivalent to $\mathscr{E}_{k_1}(p)$.
2. If $k_1 = k_3$ and k_2 is different, then the key size is doubled to $2 \cdot n$.
3. If all three keys are different the key size tripled to $3 \cdot n$.[1]

So why not applying 100 keys to increase security? The problem is the increasing time for encryption and decryption that makes it no longer practical at some point.

Clearly, if the encryption functions itself would generate a (small) group then applying the encryption resp. decryption several times would not give any advantage in terms of security. For example, take the data encryption standard (DES) encryption function (cf. Section 6.2): It has 2^{56} encryption functions, for each 56 bit key one. Still, we have $\left(2^{56}!\right)$ possible permutations. It is shown by Campell and Wiener that the set of DES encryption functions is not closed under composition, i.e., they do not build a group. In other words, there exist keys k_1 and k_2 such that $\mathsf{DES}_{k_2} \circ \mathsf{DES}_{k_1} \neq \mathsf{DES}_k$ for all keys k. It follows that the number of permutations of the form $\mathsf{DES}_{k_2} \circ \mathsf{DES}_{k_1}$ is much larger than the number of permutations of type DES_k. □

Until now we always considered that our plaintexts have the same size as the key. Clearly, in general, one wants to encrypt longer documents or texts. For this problem there are several different modes one can apply block ciphers.

3.2 Modes of Block Ciphers

Let us assume in this section that $\Sigma = \mathbb{Z}/2\mathbb{Z}$, block size is $n \in \mathbb{N}_{>0}$ and the key spaces $\mathscr{K}' = \mathscr{K}$ are the same. We switch between representations of plaintexts: For example let $n = 3$, then we can identify all natural numbers between 0 and 7. So we can represent 0 binary as 000 or $(0,0,0) \in (\mathbb{Z}/2\mathbb{Z})^3$, or 5 as 101 or $(1,0,1)$.

We further assume that there is some magic that randomly resp. pseudo randomly and uniformly distributed chooses a key $k \in \mathscr{K}$.[2]

[1] The idea of applying three different keys and not only two comes from the so-called **meet-in-the-middle attacks** which allows attacking two key encryptions with nearly the same runtime as one key encryption (but with a bigger space complexity.

[2] More on randomness later, here we keep it simple and out of our way.

Assume we have a plaintext $p \in \mathscr{P}$ of arbitrary but finite length. We divide p into blocks of length n. If the length of p is not divisible by n then we add some random symbols at the end of p. In the end we receive a repesentation $p = (p_1, \dots, p_m)$ where all p_i are plaintext blocks of length n.

Each plaintext block p_i is encrypted to a corresponding ciphertext block c_i using a given key k. In the end, the ciphertext corresponding to $p = (p_1, \dots, p_m)$ is $c = (c_1, \dots, c_m)$. The fitting decryption process works exactly the same way: One takes each block c_i and applies the decryption function $\mathscr{D}$ with the fitting key k' for k in order to receive the plaintext block p_i.

Electronic Codebook Mode (ECB)

The electronic codebook mode is the easiest mode of block ciphers. In Figure 3.1 we visualize the encryption process of the ECB mode:

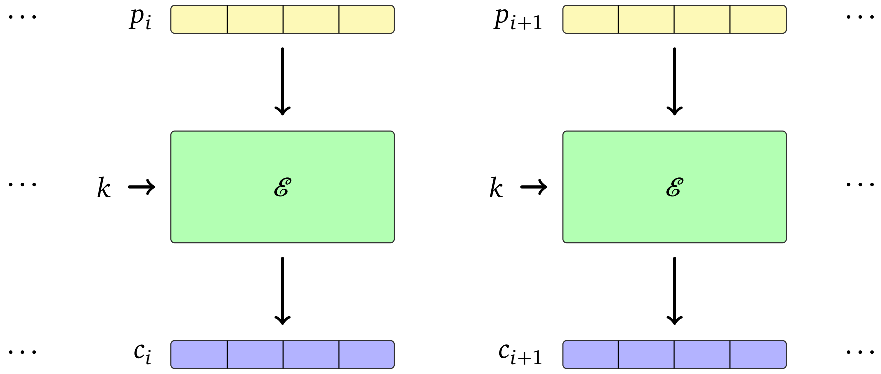


Figure 3.1: ECB encryption with block size $n = 4$

Example 3.5. Let us assume that we have $n = 3$ and we want to encrypt the plaintext $p = 1011011$. p is of length 7 the blocks look like 101 101 1, so we add zeroes to the last block until it has size 3: $p = (p_1, p_2, p_3)$ such that $p_1 = 101$, $p_2 = 101$, $p_3 = 100$. We use the permutation cipher from Example 3.3, i.e. $\mathscr{K} = \mathscr{K}' = S_3$. Let us assume the key $k = (1\,2\,3)$. We encrypt in ECB mode each block on its own:

$$\begin{aligned} c_1 &= 011 &&= \mathscr{E}_k(p_1) \\ c_2 &= 011 &&= \mathscr{E}_k(p_2) \\ c_3 &= 001 &&= \mathscr{E}_k(p_3). \end{aligned}$$

So we receive the ciphertext $c = (c_1, c_2, c_3) = 011\,011\,001$. □

Decryption in ECB mode works exactly the same way as encryption: Use the corresponding key k' and apply the corresponding decryption function $\mathscr{D}_{k'}$ to the ciphertext blocks to receive plaintext blocks:

One clearly sees the disadvantage of ECB mode: Same plaintext blocks ($p_1 = p_2$) are encrypted to the same ciphertext blocks ($c_1 = c_2$). Thus ECB mode does not hide data patterns and it is not recommended to use in cryptosystems at all anymore. For a nice visualization of the above explained problem search online for “ECB penguin”.

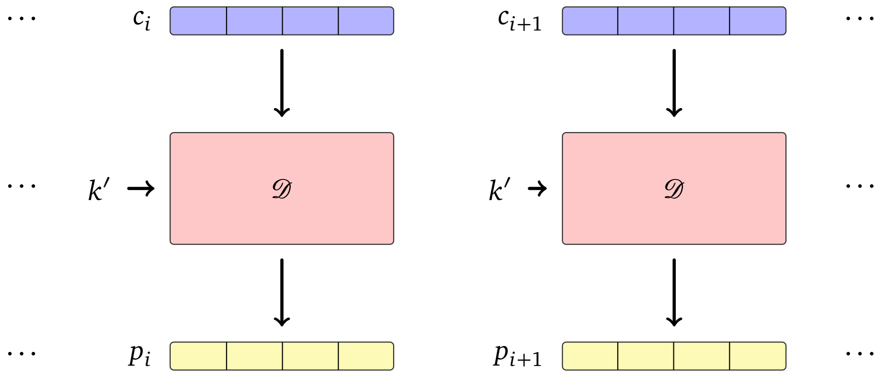


Figure 3.2: ECB decryption with block size $n = 4$

Cipherblock Chaining Mode (CBC)

Ehrsam, Meyer, Smith and Tuchman invented and patented in 1976 the cipherblock chaining mode. The main idea is to make encryption contextual: The encryption of each block depends not only on the key but also on the ciphertext of the previous block. Two question arise immediately: How does the previous ciphertext block act on the current plaintext block? How is the very first plaintext block encrypted?

Definition 3.6. The operation

$$\oplus : \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \to \mathbb{Z}/2\mathbb{Z}, (a, b) \mapsto a \oplus b$$

is defined by the following truth table:

a	b	$a \oplus b$
0	0	0
1	0	1
0	1	1
1	1	0

$\oplus$ is called **exclusive or** or **exclusive disjunction**, shortly denoted by XOR. Moreover, we extend notation to apply XOR also on elements of $(\mathbb{Z}/2\mathbb{Z})^n$ for some $n \in \mathbb{N}_{>0}$: Let $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n) \in (\mathbb{Z}/2\mathbb{Z})^n$, then we define

$$a \oplus b := (a_1 \oplus b_1, \ldots, a_n \oplus b_n).$$

□

So XOR answers the first question, the current paintext block p_i is XORed with the previous ciphertext block c_{i-1} for $i > 1$. So we need to have something to XOR p_1 with:

Definition 3.7. An **initialization vector** for a block cipher of block size n over an alphabet Σ is an element $v \in \Sigma^n$. It is randomly resp. pseudo randomly chosen[3] from Σ^n. □

[3] No, we are still not talking about this. Keep calm.

So the initialization vector IV will be the block we XOR p_1 with.

In Figures 3.3 and 3.4 we illustrate the encryption and decryption in CBC mode on block size $n = 4$ with corresponding keys k and k'.

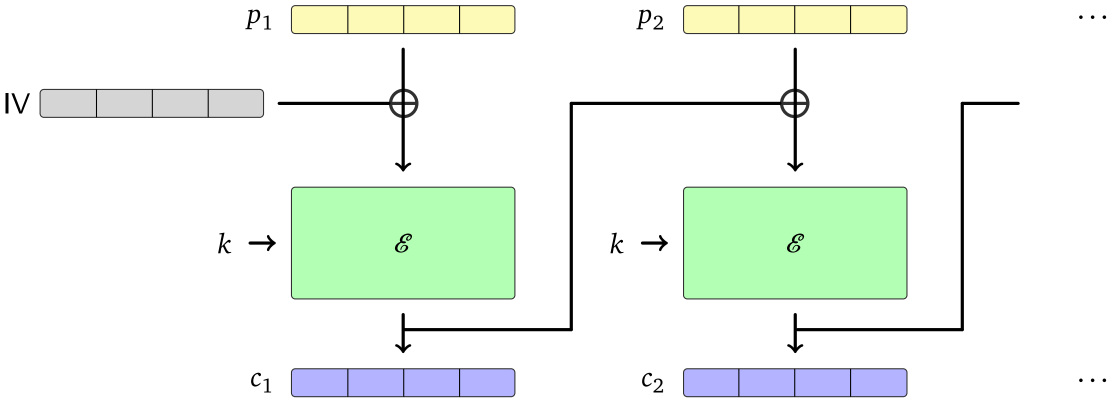


Figure 3.3: CBC encryption with block size $n = 4$

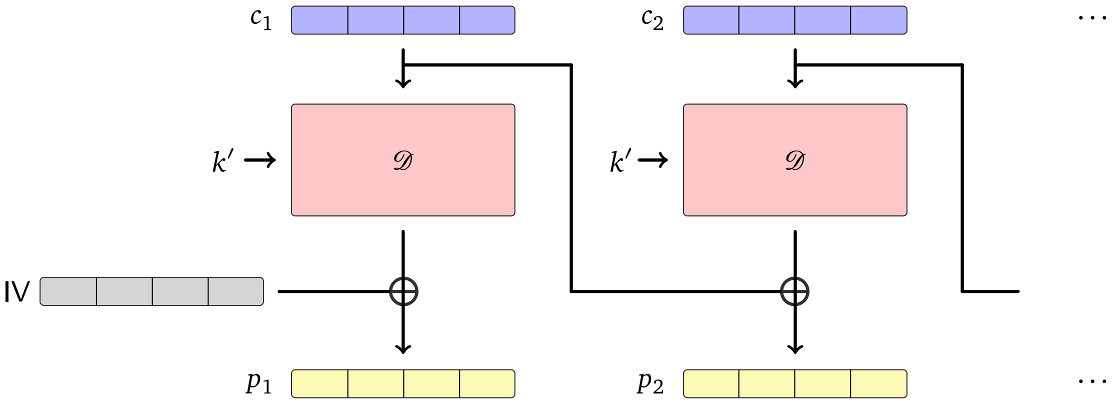


Figure 3.4: CBC decryption with block size $n = 4$

Mathematically we can formulate CBC mode in the following way:

1. For encryption we have $c_o = \mathsf{IV}$ and $c_i = \mathscr{E}_k(p_i \oplus c_{i-1})$ for $i \geq 1$.
2. For decryption we have $c_o = \mathsf{IV}$ and $p_i = \mathscr{D}_{k'}(c_i) \oplus c_{i-1}$ for $i \geq 1$.

Remark 3.8.

1. If we do not use IV for the start the first block's encryption and decryption is just done in ECB mode.
2. A change of one bit in IV or in a plaintext block affects **all** following ciphertext blocks. The problem is that due to this dependency of previous ciphertext blocks encryption in CBC mode cannot run in parallel on a computer, it must be handled sequentially.
3. Decrypting with the correct key but a wrong IV affects only the correctness of the first block of plaintext. All other blocks will be decrypted correctly. This is due to the fact that for de-

cryption we only need the previous ciphertext block, but not the decrypted previous plaintext block. It follows that decryption in CBC mode can be done in parallel.

4. Changing one bit in a ciphertext block causes complete corruption of the corresponding plaintext block, and inverts the corresponding bit in the following plaintext block. All other blocks stay correct. This fact is used by several attacks on this mode, see, for example, [Wik16e]
5. In order to make such attacks more difficult there is also a variant of CBC called PCBC: the **propagating cipherblock chaining** mode. which also takes the previous plaintext block into account when encrypting resp. decrypting the current plaintext block.

□

Example 3.9. Let us recall Example 3.5: $n = 3$, $p = 101\ 101\ 100$ and $k = (1\,2\,3)$. We choose $\mathsf{IV} = 101$.

$$\begin{array}{lllll} c_1 & = & 000 & = & \mathscr{E}_k(p_1 \oplus \mathsf{IV}) \\ c_2 & = & 011 & = & \mathscr{E}_k(p_2 \oplus c_1) \\ c_3 & = & 111 & = & \mathscr{E}_k(p_3 \oplus c_2). \end{array}$$

So we receive the ciphertext $c = (c_1, c_2, c_3) = 000\ 011\ 111$. We see that in CBC mode $c_1 \neq c_2$ whereas $p_1 = p_2$. □

Cipher Feedback Mode (CFB)

CFB mode is closely related to CBC mode. Note that decryption is done via the encryption function and almost identical to CBC encryption performed in reverse (see Figure 3.6).

1. For encryption we have $c_o = \mathsf{IV}$ and $c_i = \mathscr{E}_k(c_{i-1}) \oplus p_i$ for $i > 1$.
2. For decryption we have $c_o = \mathsf{IV}$ and $p_i = \mathscr{E}_k(c_{i-1}) \oplus c_i$ for $i > 1$.

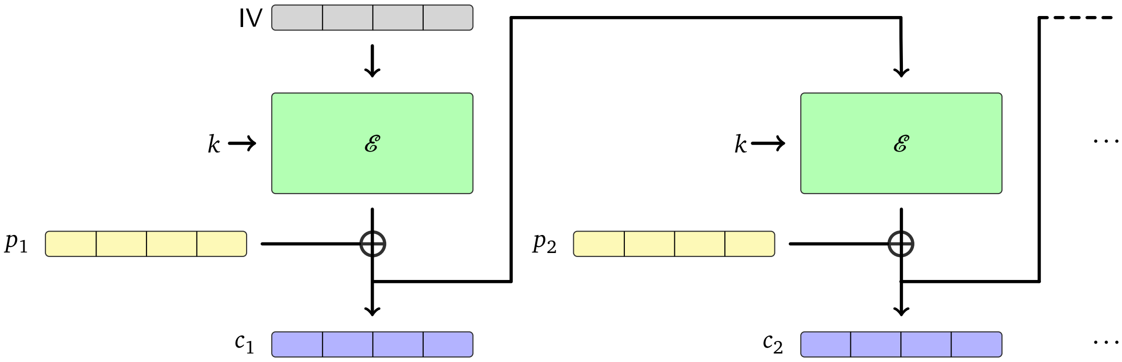


Figure 3.5: CFB encryption with block size $n = 4$

The main feature of CFB mode is that it is a so-called **self-synchronising** cipher: If some part of the ciphertext is lost the receiver only loses some parts of the plaintext, but is able to correctly decrypt other parts after some amount of input data. The errors do not propagate through the complete ciphertext. In particular, even if the input vector is unknown only the first block cannot be decrypted.

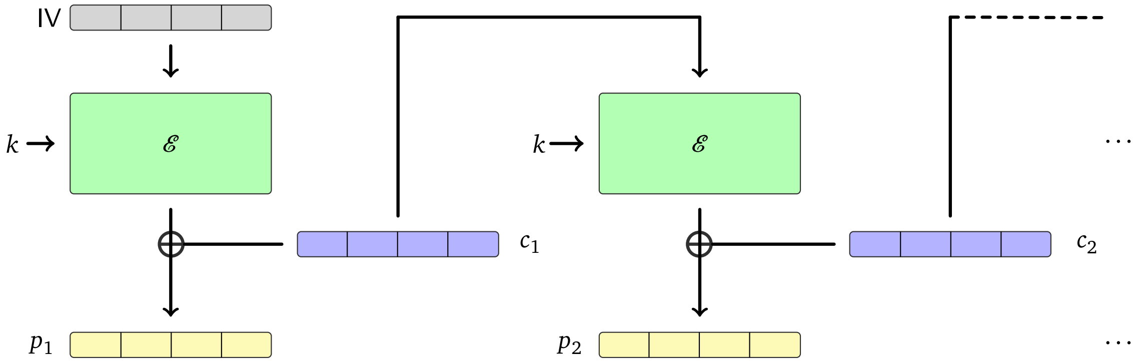


Figure 3.6: CFB decryption (using the encryption function) with block size $n = 4$

Remark 3.10.

1. Note again: Decryption is done via the encryption function $\mathscr{E}_k$.
2. Note that whereas encryption in CFB mode is recursive, i.e. it can only be handled in sequential order, decryption (see Figure 3.6) is not. Thus decryption can be parallelized.
3. CFB mode generates a so-called **stream cipher** (see Section 3.3): A continuous stream of blocks of ciphers is generated since the output of the block cipher is used as input for the next block. This generates a so-called **keystream** as input for $\mathscr{E}_k$.
4. Moreover, one can use CFB mode on plaintexts whose lengths are not multiples of the block size n by shifting through r bits in each step where $r \in \mathbb{N}_{>0}$, $r < n$ and the size of p has to be a multiple of r.

□

Example 3.11. Let us recall Example 3.9: $n = 3$, $p = 101\,101\,100$, $k = (1\,2\,3)$ and $\mathsf{IV} = 101$.

$$\begin{aligned} c_1 &= 110 &&= \mathscr{E}_k(\mathsf{IV}) \oplus p_1 \\ c_2 &= 000 &&= \mathscr{E}_k(c_1) \oplus p_2 \\ c_3 &= 100 &&= \mathscr{E}_k(c_2) \oplus p_3. \end{aligned}$$

So we receive the ciphertext $c = (c_1, c_2, c_3) = 110\,000\,100$. □

Output Feedback Mode (OFB)

OFB mode is again related to CFB mode, but it loses the property of being self-synchronising. The main difference to CFB comes after applying the encryption function $\mathscr{E}_k$: In OFB this output is taken as input for the encryption process part of the next block. In CFB mode one first XORs with the plaintext block (during encryption) resp. ciphertext block (during decryption) and uses the result as input for the encryption process part $\mathscr{E}_k$ of the next block.

In formula OFB mode can be represented in the following way:

1. We construct the input values for the encryption process $\mathscr{E}_k$ via $I_0 = \mathsf{IV}$, $O_j = \mathscr{E}_k(I_j)$ for all j. From I_0 we can then inductively generate $I_j = O_{j-1}$ for all $j > 0$.

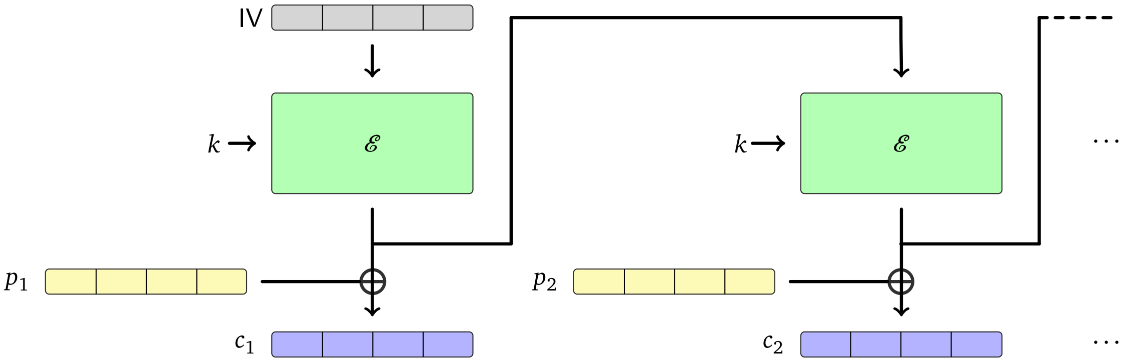


Figure 3.7: OFB encryption with block size $n = 4$

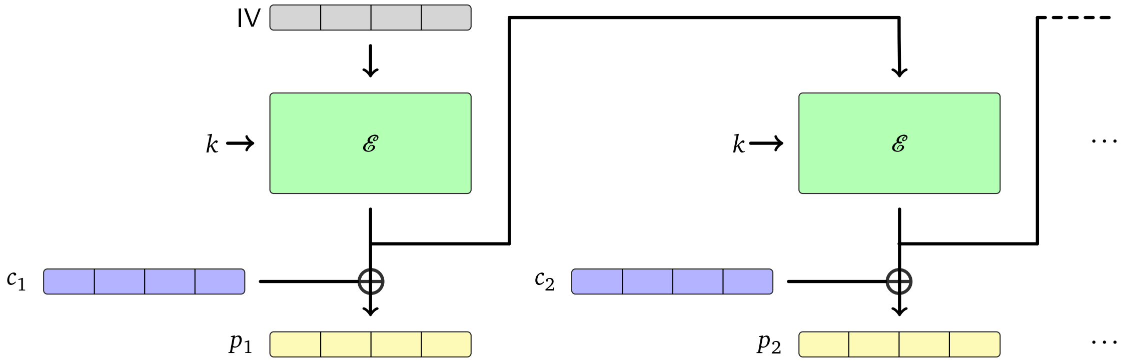


Figure 3.8: OFB decryption (using the encryption function) with block size $n = 4$

2. For encryption we have $c_j = p_j \oplus O_j$ for all $j \geq 1$.
3. For decryption we have $p_j = c_j \oplus O_j$ for all $j \geq 1$.

Remark 3.12.

1. Like in CFB mode, decryption is done via the encryption function $\mathscr{E}_k$.
2. In contrast to CFB mode the cipher in OFB mode is not self-synchronising. For example, if the input vector IV is lost during transmission this error has impact on the decryption of all blocks. On the other hand, an error in a bit of some ciphertext block only appears in the corresponding block of the plaintext and does not affect any other blocks.
3. Like CFB mode, also OFB mode generates a so-called **stream cipher** (see Section 3.3). In contrast to CFB mode, the generated keystream is independent of the plaintext resp. ciphertext blocks.
4. Even more, the keystream depends only on the intialization vector IV. Thus, in contrast to CFB mode, for each new communication a new initialization vector IV has to be chosen (randomly!).
5. Like CFB mode, OFB mode can be applied to plaintexts whose lengths are not multiples of the block size n.

6. In OFB mode one can parallelize encryption and decryption partly: The idea is to first take the initialization vector IV and apply to it $\mathcal{E}_k$. This is the input of the application of $\mathcal{E}_k$ for the next block. So one can sequentially precompute all these intermediate blocks, and then XOR them, in parallel, with the corresponding plaintext block (when encrypting) resp. the corresponding ciphertext block (when decrypting).

7. By construction, the first ciphertext block in CFB and OFB mode is encrypted equivalently.

□

Example 3.13. Let us recall Example 3.11: $n = 3$, $p = 101\ 101\ 100$, $k = (1\,2\,3)$ and $\mathrm{IV} = 101$.

$$\begin{array}{rclcl} O_1 & = & 011 & = & \mathcal{E}_k(\mathrm{IV}) \\ O_2 & = & 110 & = & \mathcal{E}_k(O_1) \\ O_3 & = & 101 & = & \mathcal{E}_k(O_2) \\ c_1 & = & 110 & = & O_1 \oplus p_1 \\ c_2 & = & 011 & = & O_2 \oplus p_2 \\ c_3 & = & 001 & = & O_3 \oplus p_3. \end{array}$$

So we receive the ciphertext $c = (c_1, c_2, c_3) = 110\ 011\ 001$. □

Counter Mode (CTR)

CTR mode is different from CBC, CFB and OFB in the sense that instead of an initialization vector it uses a so-called **nonce**:

Definition 3.14. A **nonce** is an arbitrary number used only once for cryptographic communication. They are often pseudo random resp. random numbers.[4] □

As we can see in Figures 3.9 and 3.10 the name counter mode comes from the fact that the nonce is combined with a counter. This can be an arbitrary counter, usually one uses an increment counter: So for each block of the plaintext or ciphertext the counter is incremented by 1. There are now various possible ways to combine the nonce with the counter, for example, they can be concatenated, added or XORed.

Let us denote by NC_i the combination of the chosen nonce with the ith counter. We can then formulate the processes in CTR mode in following way:

1. For encryption we have $c_j = p_j \oplus \mathcal{E}_k(\mathrm{NC}_j)$ for all j.

2. For decryption we have $p_j = c_j \oplus \mathcal{E}_k(\mathrm{NC}_j)$ for all j.

Remark 3.15.

1. Like OFB mode, CTR mode generates a stream cipher with a keystream. Also like OFB, the keystream can be precomputed.

2. Like OFB mode, in CTR mode errors in bits of ciphertext blocks affect only the corresponding plaintext block, but no other blocks.

3. Encryption and decryption of the blocks can be done in parallel.

[4] Still nothing here.

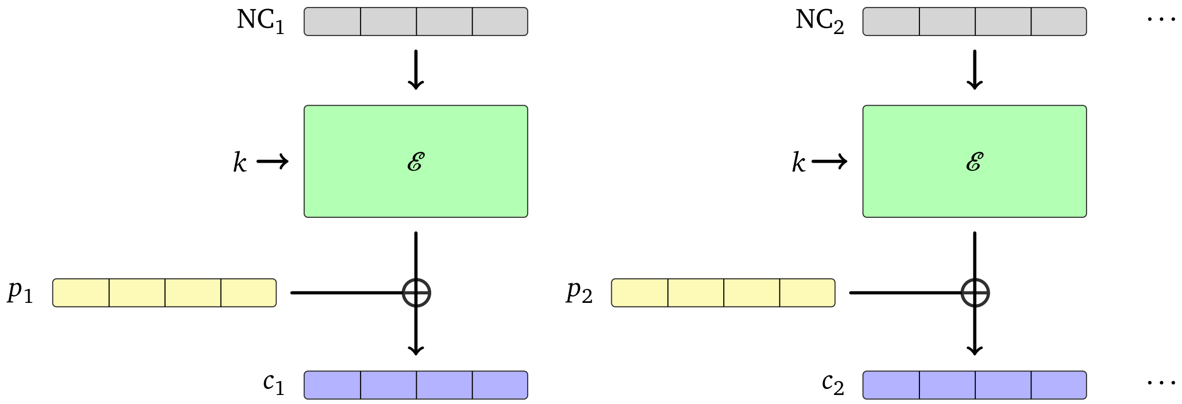


Figure 3.9: CTR encryption with block size $n = 4$

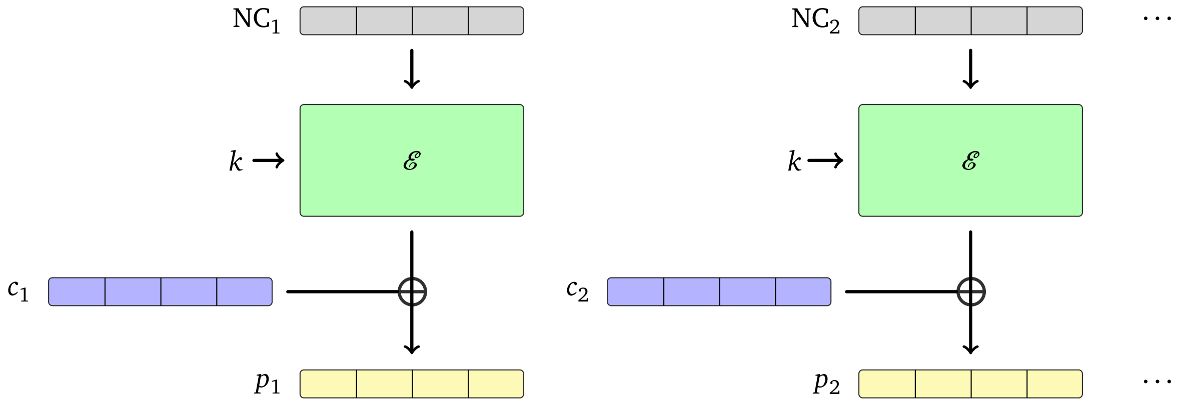


Figure 3.10: CTR decryption (using the encryption function) with block size $n = 4$

4. The nonce should not be reused, there are many attacks exploiting reused nonces in order to get information about the keys, like CCA may then be applicable.

□

Example 3.16. Let us recall Example 3.13: $n = 3$, $p = 101\ 101\ 100$, $k = (1\,2\,3)$. This time take as nonce the random number 5, in binary representation 101. As counter we use the usual increment. We combine the nonce with the counter by addition.

$$\begin{array}{lllll}
\mathrm{NC}_1 & = & 011 & = & \mathscr{E}_k(101) \\
\mathrm{NC}_2 & = & 101 & = & \mathscr{E}_k(110) \\
\mathrm{NC}_3 & = & 111 & = & \mathscr{E}_k(111) \\
c_1 & = & 110 & = & \mathrm{NC}_1 \oplus p_1 \\
c_2 & = & 000 & = & \mathrm{NC}_2 \oplus p_2 \\
c_3 & = & 011 & = & \mathrm{NC}_3 \oplus p_3.
\end{array}$$

So we receive the ciphertext $c = (c_1, c_2, c_3) = 110\ 000\ 011$. □

Note that until now we are only thinking about encryption and decryption. The above modes can also be combined with authentication procedures. For example, CTR mode together with authentication is known as **Galois Counter mode** (GCM). Its name comes from the fact that the authentication function is just a multiplication in the Galois field $\mathrm{GF}(2^k)$ for some $k \in \mathbb{N}$. For example, the AES cryptosystem (see Section 6.3) uses $k = 128$ by default. We will discuss authentication later on, for example, see Chapter 13.

Nowadays, CBC and CTR mode are widely accepted in the crypto community. For example, they are used in encrypted ZIP archives (also in other archiving tools), CTR is used in TLS 1.2[5] in your web browser or for off-the-record messaging (OTR)[6] in your messaging app (usually in combination with AES).

3.3 Stream Ciphers

In contrast to block ciphers, stream ciphers can handle plaintexts resp. ciphertext of any size without the necessity of having blocks.

Definition 3.17. Let Σ be an alphabet.

1. A **keystream** is a stream of random or pseudo random[7] characters or digits from Σ that are combined with a plaintext (ciphertext) digit in order to produce a ciphertext (plaintext) digit.
2. A **stream cipher** is a symmetric cipher acting on plaintexts and ciphertexts $\mathscr{P}, \mathscr{C} \subset \Sigma^\bullet$ of any given length. Each plaintext digit (e.g. binary representation) is encrypted one at a time with a corresponding digit of a keystream.
3. A stream cipher is called **synchronous** if the keystream is independent of the plaintext and the ciphertext.
4. A stream cipher is called **self-synchronizing** if the keystream is computed depending on the n previous ciphertext digits for some $n \in \mathbb{N}_{>0}$.

□

Example 3.18. CFB mode and OFB mode are examples of stream ciphers. CFB is self-synchronizing whereas OFB is synchronous. □

Remark 3.19.

1. In practice a digit is usually a bit (binary representation) and the combination is just XOR.
2. Self-synchronizing stream ciphers have the advantage that if some ciphertext digits are lost or corrupted during transmission this error will not propagate to all following ciphertext digits but only to the next n ones. In a synchronous stream cipher one cannot recover a ciphertext digit lost or added during transmission, the complete following decryption may fail.

□

[5] TLS stands for **Transport Layer Security**. It is a protocol that provides private resp. encrypted communication, authentication and integrity for a lot of applications like web browsing, emails, messaging, voice-over-IP, etc.

[6] OTR provides deniable authentication, i.e. after an encrypted communication no one can ensure that a given encryption key was used by a chosen person.

[7] We really will come to this topic soon!

Binary stream ciphers are usually implemented using so-called linear feedback shift registers:

Definition 3.20. Let $\Sigma = \mathbb{Z}/2\mathbb{Z}$, $\mathscr{P}, \mathscr{C} \subset \Sigma^\bullet$, $\mathscr{K}' = \mathscr{K} = \Sigma^n$ for some $n \in \mathbb{N}_{>0}$. Let $\mathsf{IV} \in \Sigma^n$ be an initialization vector. Assume a plaintext $p = p_1 \dots p_m$ of some size m in $\Sigma^\bullet$. The keystream $k = k_1 k_2 k_3 \dots$ is initialized via

$$k_i = \mathsf{IV}_i \text{ for all } 1 \leq i \leq n.$$

A **linear feedback shift register** (LFSR) is a shift register used for generating pseudo random bits. Its input bit is a linear function f of its previous state. For $j > 0$ one now generates

$$k_{j+n} = f(k_j, \dots, k_{j+n-1}).$$

Encryption is now done via $\mathscr{E}_k(p) = p_1 \oplus k_1 \dots, p_m \oplus k_m = c$, Analogously, decryption is performed with $\mathscr{D}_k(c) = c_1 \oplus k_1, \dots, c_m \oplus k_m = p$. □

In general, this linear function is XOR resp. a combination of several XORs.

Example 3.21. Let $\Sigma = \mathbb{Z}/2\mathbb{Z}$, $n = 3$ and $\mathsf{IV} = 110$. As linear feedback function we use

$$k_{j+3} = k_{j+2} + k_j \mod 2 \text{ for all } j \geq 1.$$

Thus, we receive the keystream

$$k = 1101001\ 110100\dots$$

So after 7 bits the keystream recurs. □

We can see in Example 3.21 that there is periodicity in the keystream. This is one of the problems of linear feedback shift registers: There is always a periodicity which contradicts the fact that the keystream should produce pseudo random bits. Thus, one tries to implement linear feedback shift registers that have a rather big periodicity. If the periodicity is too small an attacker can easily recover the keystream.

Another possible attack is a KPA attack: If an attacker has a plaintext and a corresponding ciphertext the keystream can be reconstructed. If the keystream has a small periodicity the attacker knows the full keystream and all following transmissions using the same keystream can be decrypted by the attacker. This is one problem of the WEP (wired equivalent privacy) algorithm that was used for a long time to encrypt wireless networks (WLAN). Nowadays a WEP key can be cracked in under a minute with a usual personal computer.

Remark 3.22. One ingredient of WEP protocol (and also the WPA protocol) is the stream cipher RC4 invented in 1987 by Ron Rivest. This stream cipher was very popular in the last decades and also used in the SSL and TLS standards for web encryption and authentication until February 2015. At this point too many attacks were known and RC4 is believed to be completely broken. All bigger web browsers like Chrome(Chromium), Firefox, Internet Explorer (Edge), Opera, Safari, etc. have removed RC4 support since December 2015 resp. January 2016. Whereas newer versions of TLS do not support RC4, many web servers still only support older versions of TLS and so the web browser fell back to using RC4. This was a security problem. □

So, if there are so many problems with the security of stream ciphers, why use them? The answer comes from the practical perspective: For stream ciphers one has very efficient software implementations, making the encryption/decryption process fast. Moreover, one can even do hardware

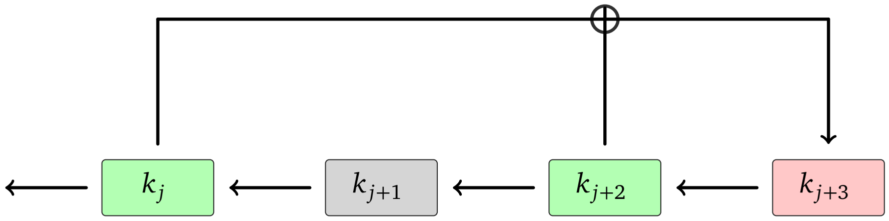


Figure 3.11: Linear feedback shift register for Example 3.21

implementations that are even faster. For example, linear feedback shift registers can be easily implemented in hardware:

In Figure 3.11 we can see four hardware registers holding the last four values of the keystream. These values are shifting to the left and a new last value is computing via $k_i \oplus k_{i+2}$.

Nowadays many cryptographers are working on new, more secure **and** fast stream ciphers.

3.4 A Short Review of Historical Ciphers

Having discussed different ciphers in detail, let us come back to the historical crpytosystems we already know. We try to investigate a bit on their security:

In Example 2.19 we have seen that both, Caesar's and Vigenère's cryptosystems are symmetric ones, they are even block ciphers. Caesar's cipher is just a special case of Vigenère's: Taking the key length to be $|k| = 1$ then we get Caesar's cipher.

Another idea for a cryptosystem that is related to the above ones comes from Lester S. Hill in 1929: Hill's cipher uses $\Sigma_1 = \Sigma_2 = \Sigma = \mathscr{P} = \mathscr{C} \cong \mathbb{Z}/m\mathbb{Z}$ (for the usual latin alphabet take $m = 26$) and $\mathscr{K} = \mathscr{K}' = \text{Mat}(n \times n, \mathbb{Z}/m\mathbb{Z})$ such that $\gcd(\det(A), m) = 1$ for $A \in \mathscr{K}$. Encryption of a plaintext block $p \in \Sigma^n$ of size n with a key $A \in \mathscr{K}$ is now done via

$$\mathscr{E} : \mathscr{K} \times (\mathbb{Z}/m\mathbb{Z})^n \to (\mathbb{Z}/m\mathbb{Z})^n, \ \mathscr{E}_A(p) = Ap \mod m.$$

The gcd condition ensures that for any $A \in \mathscr{K}$ there exists an A^{-1} that can be used as key for decryption.

One can easily see Hill's cipher is the most general linear block cipher. One can show that the permutation cipher (see Example 3.3) is a special case:

Example 3.23. As in Example 3.3 we assume $\mathscr{K} = \mathscr{K}' = S_n$. Let e_i be the ith unit vector in Σ^n for $1 \leq i \leq n$. If we now choose a permutation $\pi \in \mathscr{K}$ and apply it to e_i we get $\mathscr{E}_\pi(e_i) = e_{\pi(i)}$. Taking each such permutation of a unit vector as a column in an $n \times n$ matrix we receive $A_\pi = \left(e_{\pi(1)}, \ldots, e_{\pi(n)}\right)$. Now we can rewrite the encryption of an arbitrary plaintext block $p = (p_1, \ldots, p_n) \in \Sigma^n$ w.r.t. π via:

$$\mathscr{E}_\pi(p) = (p_{\pi(1)}, \ldots, p_{\pi(n)}) = A_\pi p.$$

Thus, the permutation cipher is a linear cipher. □

Next one could generalize the above ideas even further, taking Hill's linear cipher together with Vigenère's shift cipher: In the end we receive an affine linear cipher:

Definition 3.24. Let $\Sigma = \mathbb{Z}/m\mathbb{Z}$ for some $m \in \mathbb{N}_{>0}$, let $\mathscr{P} = \mathscr{C} = \Sigma^n$ for $n \in \mathbb{N}_{>0}$. Moreover, let $\mathscr{K} = \mathscr{K}' = \mathrm{Mat}(n \times n, \Sigma) \times \Sigma^n$ such that for $k = (A, b) \in \mathscr{K}$ it holds that $\gcd(\det(A), m) = 1$. An **affine linear block cipher** is then defined by the encryption function $\mathscr{E}_k$ where $k = (A, b)$ applied to the plaintext block $p = (p_1, \dots, p_n) \in \Sigma^n$ such that

$$\mathscr{E}_k : \Sigma^n \to \Sigma^n, \ \ p \mapsto Ap + b \mod m.$$

The corresponding decryption with $k' = (A^{-1}, b)$ is done via

$$\mathscr{D}_{k'} : \Sigma^n \to \Sigma^n, \ \ c \mapsto A^{-1}(c - b) \mod m.$$

□

Clearly, all other considered ciphers are affine linear ciphers:

1. For Caesar's cipher take $n = 1$, $\mathscr{K} = \mathscr{K}' = \{E_n\} \times \Sigma$ where E_n is the identity matrix.
2. For Vigenère's cipher take $n \in \mathbb{N}_{>0}$, $\mathscr{K} = \mathscr{K}' = \{E_n\} \times \Sigma$.
3. For Hill's cipher take $n \in \mathbb{N}_{>0}$, $\mathscr{K} = \mathscr{K}' = \mathrm{Mat}(n \times n, \Sigma) \times \{0\}$.

Let us finish this section by trying to understand why all these ciphers based on affine linear functions are broken nowadays:

We assume an arbitrary affine linear cipher with $\Sigma = \mathbb{Z}/m\mathbb{Z}$ for some $m \in \mathbb{N}_{>0}$, let $\mathscr{P} = \mathscr{C} = \Sigma^n$ for $n \in \mathbb{N}_{>0}$. Moreover, let $\mathscr{K} = \mathscr{K}' = \mathrm{Mat}(n \times n, \Sigma) \times \Sigma^n$ such that for $k = (A, b) \in \mathscr{K}$ it holds that $\gcd(\det(A), m) = 1$. Encryption is then given via the encryption function $\mathscr{E}_k$ where $k = (A, b)$ applied to a plaintext block $p \in \Sigma^n$ such that

$$\mathscr{E}_k : \Sigma^n \to \Sigma^n, \ \ p \mapsto Ap + b \mod m.$$

We try to do a KPA, so the attacker has $n + 1$ plaintext blocks $p_0, p_1, \dots, p_n \in \Sigma^n$ and the corresponding ciphertext blocks $c_0, c_1, \dots, c_n \in \Sigma^n$. Then it holds that

$$c_i - c_0 \equiv A(p_i - p_0) \mod m \text{ for all } 1 \le i \le n. \tag{3.1}$$

We construct two matrices:

1. Matrix P whose n columns are the differences $p_i - p_0 \mod m$ for $1 \le i \le n$.
2. Matrix C whose n columns are the differences $c_i - c_0 \mod m$ for $1 \le i \le n$.

Thus we can rewrite Equation 3.1 with matrices:

$$C \equiv AP \mod m.$$

If $\gcd(\det(P), m) = 1$, we can compute P^{-1} and thus recover A:

$$A \equiv CP^{-1} \mod m.$$

Once we know A we can also recover b:

$$b \equiv c_0 - Ap_0 \mod m.$$

Thus we have found the key $k = (A, b)$. If the cipher is linear, i.e. $b = 0$ then one can set $p_0 = c_0 = 0 \in \Sigma^n$ and we can find A even with only n plaintext-ciphertext block pairs.

Chapter 4

Information Theory

In the last chapter we have seen that all affine linear ciphers are broken resp. not secure nowadays. Thus the question arises if there exist cryptosystems that are provable secure in a mathematical sense. It turns out that this is possible.

4.1 A Short Introduction to Probability Theory

In order to discuss the security of a cryptosystem we need some basic knowledge of probability theory.

Definition 4.1. Let Ω be a finite nonempty set and $\mu : \Omega \to [0,1]$ be a map with $\sum_{x\in\Omega} \mu(x) = 1$. For $A \subset \Omega$ we define $\mu(A) = \sum_{x\in A} \mu(x)$.

1. μ is called a **probability distribution**.

2. The tuple (Ω, μ) is called a **finite probability space**.

3. $A \subset \Omega$ is called an **event**, an element $x \in \Omega$ is called an **elementary event**.

4. The probability distribution $\bar{\mu}$ defined by $\bar{\mu}(x) := \frac{1}{|\Omega|}$ is called the **(discrete) uniform distribution** on Ω.

5. Let $A, B \subset \Omega$ such that $\mu(B) > 0$. Then we define the **conditional probability**

$$\mu(A|B) := \frac{\mu(A \cap B)}{\mu(B)}.$$

   That is, the probability of A given the occurence of B.

6. $A, B \subset \Omega$ are called **(statistically) independent** if

$$\mu(A \cap B) = \mu(A)\mu(B).$$

□

Lemma 4.2. Let (Ω, μ) be a finite probability space and $A, B \subset \Omega$ be events.

1. $\mu(\emptyset) = 0$, $\mu(\Omega) = 1$.

2. $\mu(\Omega \setminus A) = 1 - \mu(A)$.

3. If $A \subset B$ then $\mu(A) \leq \mu(B)$.
4. $\mu(A \cap B) = \mu(A|B)\mu(B)$.

□

Proof. All statements follow from Definition 4.1. ■

For the conditional probability there is the well-known formula by Thomas Bayes:

Theorem 4.3 (Bayes)**.** Let (Ω, μ) be a finite probability space and $A, B \subset \Omega$ be two events such that $\mu(A), \mu(B) > 0$. Then

$$\mu(A|B) = \mu(B|A)\frac{\mu(A)}{\mu(B)}.$$

□

Proof. By definition we have $\mu(A|B) = \frac{\mu(A\cap B)}{\mu(B)}$ and $\mu(B|A) = \frac{\mu(A\cap B)}{\mu(A)}$. Thus we conclude

$$\mu(A|B)\mu(B) = \mu(A \cap B) = \mu(B|A)\mu(A).$$

■

Definition 4.4. Let (Ω, μ) be a finite probability space.

1. Let M be a set. A map $X : \Omega \to M$ is called an **(M-valued discrete) random variable** on Ω.
2. Let M be some set and let X be an M-valued random variable.

   a) The **distribution of** X is defined by

   $$\mu_X(m) := \mu(X = m) := \mu\left(X^{-1}(m)\right) \text{ for all } m \in M.$$

   More general, for $A \subset M$ we define

   $$\mu_X(A) := \mu(X \in A) := \mu\left(X^{-1}(A)\right).$$

   b) X is called **uniformly distributed** if

   $$\mu_X(m) = \frac{1}{|M|} \text{ for all } m \in M.$$

3. If $M \subset \mathbb{C}$ and X is an M-valued random variable, then we define the **expected value of** X by

   $$E(X) := \sum_{x\in\Omega} X(x)\mu(x) \in \mathbb{C}.$$

   Moreover, let Y be another M-valued random variable. We define

   $$\begin{array}{rlrcl} X+Y: & \Omega \to \mathbb{C}, & (X+Y)(x) & = & X(x)+Y(x), \\ XY: & \Omega \to \mathbb{C}, & (XY)(x) & = & X(x)\cdot Y(x). \end{array}$$

4. Let $X_i : \Omega \to M_i$ be random variables for sets M_i for $1 \leq i \leq n$. For $m_i \in M_i$ we define the **product probability distribution**

   $$\mu_{X_1,\ldots,X_n}(m_1,\ldots,m_n) := \mu(X_1 = m_1,\ldots,X_n = m_n) := \mu\left(\cap_{i=1}^n \{X_i = m_i\}\right).$$

Let $X : \Omega \to M$ and $Y : \Omega \to N$ be two random variables for sets M, N.

5. For $\mu_Y(n) > 0$ we define the **conditional probability** of $X = m$ given the occurrence of $Y = n$ by
$$\mu_{X|Y}(m|n) := \frac{\mu_{X,Y}(m,n)}{\mu_Y(n)}.$$

6. X and Y are called **(statistically) independent** if
$$\mu_{X,Y}(m,n) = \mu_X(m)\mu_Y(n) \text{ for all } m \in M, n \in N.$$

□

The following properties are easily checked with the above definitions.

Lemma 4.5. Let (Ω, μ) be a finite probability space,

1. Let M, N be sets and let $X : \Omega \to M$ and $Y : \Omega \to N$ be two random variables.
   a) (Bayes' formula) For $\mu_X(m), \mu_Y(n) > 0$ it holds that
   $$\mu_{X|Y}(m|n) = \mu_{Y|X}(n|m)\frac{\mu_X(m)}{\mu_Y(n)}.$$
   b) X and Y are independent iff $\mu_Y(n) = 0$ or $\mu_{X|Y}(m|n) = \mu_X(m)$ for all $m \in M$, $n \in N$.
2. Let $X, Y : \Omega \to \mathbb{C}$ be two random variables. Then the following statements hold:
   a) $E(X) = \sum_{m \in \mathbb{C}} m\mu_X(m)$.
   b) $E(X + Y) = E(X) + E(Y)$.
   c) $E(XY) = E(X) \cdot E(Y)$ if X and Y are independent. The converse is false.

□

Proof. Exercise. ■

Example 4.6. A nice example of probability theory is the so-called *birthday paradox*: How many people have to be at a party such that the probability of two having birthday on the very same day is at least $1/2$?

We can look at this in a more general fashion: Assume that there are n different birthdays and we have k people at the party. An elementary element is a tupel $(g_1, \ldots, g_k) \in \{1, 2, \ldots, n\}^k$. If this elementary element takes place then the ith person has birthday g_i for $1 \leq i \leq k$. All in all, if A denotes the set of events we have $|A| = n^k$. We assume a uniform distribution, i.e. each elementary element has probability $\frac{1}{n^k}$.

At least two people should have birthday on the same day, denote this probability by P. Thus $Q = 1 - P$ is the probability that all have different birthdays. Q is easier to compute, so we do this: For each $(g_1, \ldots, g_k)$ it should hold that $g_i \neq g_j$ for $i \neq j$. Let us call the event consisting of all these elementary elements B. For each such element in $\{1, 2, \ldots, n\}^k$ we have n choices for the first entry, $n-1$ choices for the second entry, ... It follows that $|B| = \prod_{i=0}^{k-1}(n-i)$. For the probability Q we have to multiply by $\frac{1}{n^k}$:

$$Q = \frac{1}{n^k}\prod_{i=0}^{k-1}(n-i) = \prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right).$$

For $x \in \mathbb{R}$ we know that $1 + x \leq e^x$, thus we can estimate Q via

$$Q \leq \prod_{i=1}^{k-1} e^{-\frac{i}{n}} = e^{-\sum_{i=1}^{k-1}\frac{i}{n}} = e^{\frac{-k(k-1)}{2n}}.$$

Thus, in order to get $Q \leq \frac{1}{2}$ (i.e. $P \geq \frac{1}{2}$) we have to choose

$$k \geq \frac{\left(1+\sqrt{1+8n\log 2}\right)}{2}.$$

So for $n = 365$ we need $k = 23$ people which is a bit more than $\sqrt{365}$, or in general $\sqrt{n}$. □

Until the end of this chapter we assume the following:

Convention 4.7. Let Π be a symmetric cryptosystem and let $\mu_{\mathcal{K}}$ be a probability distribution on the keyspace $\mathcal{K} = \mathcal{K}'$. We assume the following

1. $\mathcal{P}, \mathcal{K}, \mathcal{C}$ are finite sets. Since $\mathcal{E}_e$ is injective by Exercise 6 we also have $|\mathcal{P}| \leq |\mathcal{C}|$.
2. $\mu_{\mathcal{K}}(e) > 0$ for all $e \in \mathcal{K}$.
3. $\mathcal{E}$ is a family of maps, i.e. not multivalued maps.
4. We define $\Omega := \mathcal{P} \times \mathcal{K}$, a set of events with elementary events (p, e) when the plaintext $p \in \mathcal{P}$ is encrypted with the key $e \in \mathcal{K}$.
5. The map $\Omega \to \mathcal{C}$, $(p, e) \mapsto \mathcal{E}_e(p)$ is surjective.
6. Any probability distribution $\mu_{\mathcal{P}}$ on $\mathcal{P}$ defines a probability distribution on Ω via
$$\mu(p,e) := \mu((p,e)) := \mu_{\mathcal{P}}(p)\mu_{\mathcal{K}}(e).$$
7. In the same way, we identify $\mathcal{P}$ and $\mathcal{K}$ with corresponding random variables by overloading notation: We denote the projection maps resp. random variables
$$\begin{array}{rlll} \mathcal{P}: & \Omega \to \mathcal{P}, & (p,e) & \mapsto & p, \\ \mathcal{K}: & \Omega \to \mathcal{K}, & (p,e) & \mapsto & e. \end{array}$$
8. The random variables $\mathcal{P}$ and $\mathcal{K}$ are independent.[1]
9. As above (again by overloading notation), we also define the random variable
$$\mathcal{C}: \quad \Omega \to \mathcal{C}, \quad (p,e) \mapsto \mathcal{E}_e(p).$$

[1] The choice of the encryption key is independent of the chosen plaintext.

Its probability distribution is then given by

$$\mu_{\mathscr{C}}(c) = \sum_{\substack{(p,e)\in\Omega\\ \mathscr{E}_e(p)=c}} \mu(p,e) \quad \text{for all } c \in \mathscr{C}.$$

□

Example 4.8. Let $\mathscr{P} = \{a, b\}$ with $\mu_{\mathscr{P}}(a) = \frac{1}{4}$ and $\mu_{\mathscr{P}}(b) = \frac{3}{4}$. Let $\mathscr{K} = \{e_1, e_2, e_3\}$ with $\mu_{\mathscr{K}}(e_1) = \frac{1}{2}$ and $\mu_{\mathscr{K}}(e_2) = \mu_{\mathscr{K}}(e_3) = \frac{1}{4}$. Let $\mathscr{C} = \{1, 2, 3, 4\}$ and let $\mathscr{E}$ be given by the following **encryption matrix**:

$\mathscr{E}$	a	b
e_1	1	2
e_2	2	3
e_3	3	4

Then we can compute the probability distribution

$$\begin{array}{lclclcl}
\mu_{\mathscr{C}}(1) & = & \mu(a,e_1) & = & \frac{1}{4}\frac{1}{2} & = & \frac{1}{8}\\
\mu_{\mathscr{C}}(2) & = & \mu(a,e_2)+\mu(b,e_1) & = & \frac{1}{4}\frac{1}{4}+\frac{3}{4}\frac{1}{2} & = & \frac{7}{16}\\
\mu_{\mathscr{C}}(3) & = & \mu(a,e_3)+\mu(b,e_2) & = & \frac{1}{4}\frac{1}{4}+\frac{3}{4}\frac{1}{4} & = & \frac{1}{4}\\
\mu_{\mathscr{C}}(4) & = & \mu(b,e_3) & = & \frac{3}{4}\frac{1}{4} & = & \frac{3}{16}.
\end{array}$$

For the conditional probability we get, for example,

$$\begin{aligned}
\mu_{\mathscr{P}|\mathscr{C}}(a|1) &= \frac{\mu_{\mathscr{P},\mathscr{C}}(a,1)}{\mu_{\mathscr{C}}(1)}\\
&= \frac{\mu(\{(p,e)=1\}\cap\{p=a\})}{\frac{1}{8}}\\
&= \mu(\{(a,e_1)\}\cap\{(a,e_1),(a,e_2),(a,e_3)\})\cdot 8\\
&= \mu(\{(a,e_1)\})\cdot 8\\
&= 8\mu_{\mathscr{P}}(a)\mu_{\mathscr{K}}(e_1)\\
&= 8\cdot\frac{1}{4}\cdot\frac{1}{2} = 1
\end{aligned}$$

That is clear: The probability of having $p = a$ when we have $c = 1$ is 1. From Bayes' formula in Lemma 4.5 we directly get

$$\mu_{\mathscr{C}|\mathscr{P}}(1|a) = \frac{\mu_{\mathscr{C}}(1)}{\mu_{\mathscr{P}}(a)}\mu_{\mathscr{P}|\mathscr{C}}(a|1) = \frac{1}{8}\cdot 4\cdot 1 = \frac{1}{2}.$$

The probability of getting $c = 1$ under the assumption of having $p = a$ is $\frac{1}{2}$ since $\mu_{\mathscr{K}}(e_1) = \frac{1}{2}$. □

4.2 Perfect Secrecy

With these settings we can now define what is known as **perfect secrecy**:

Definition 4.9 (Shannon)**.** A cryptosystem Π is called **perfectly secret for** $\mu_{\mathscr{P}}$ if $\mathscr{P}$ and $\mathscr{C}$ are independent, that is

$$\begin{array}{llll} \forall p \in \mathscr{P}, c \in \mathscr{C}: & \mu_{\mathscr{P}}(p) = 0 & \text{or} & \mu_{\mathscr{C}|\mathscr{P}}(c|p) = \mu_{\mathscr{C}}(c) \text{ (or, equivalently,)} \\ \forall p \in \mathscr{P}, c \in \mathscr{C}: & \mu_{\mathscr{C}}(c) = 0 & \text{or} & \mu_{\mathscr{P}|\mathscr{C}}(p|c) = \mu_{\mathscr{P}}(p). \end{array}$$

We call Π **perfectly secret** if Π is perfectly secret for any probability distribution $\mu_{\mathscr{P}}$. □

Example 4.10. The cryptosystem from Example 4.8 is not perfectly secret, since it is not perfectly secret for the given $\mu_{\mathscr{P}}$: For $a \in \mathscr{P}$ and $1 \in \mathscr{C}$ we have $\mu_{\mathscr{P}}(a) = \frac{1}{4} \neq 0$ and $\mu_{\mathscr{C}|\mathscr{P}}(1|a) = \frac{1}{2} \neq \frac{1}{8} = \mu_{\mathscr{C}}(1)$. □

Remark 4.11. Perfect secrecy of a cryptosystem Π means that the knowledge of a ciphertext $c \in \mathscr{C}$ does not yield **any** information on the plaintext $p \in \mathscr{P}$. □

The next properties are essential for perfect secret cryptosystems as we see in Lemma 4.14 and Theorem 4.22.

Definition 4.12. Let Π be a cryptosystem. We call Π resp. $\mathscr{E}$ **transitive / free / regular** if for each $(p, c) \in \mathscr{P} \times \mathscr{C}$ there is one / at most one / exactly one $e \in \mathscr{K}$ such that $\mathscr{E}_e(p) = c$. □

Lemma 4.13. Let Π be a cryptosystem. For each $p \in \mathscr{P}$ let $\tilde{p} : \mathscr{K} \to \mathscr{C}$ be the map defined by $e \mapsto \mathscr{E}_e(p)$. Then it holds:

1. $\mathscr{E}$ is transitive iff for all $p \in \mathscr{P}$ $\tilde{p}$ is surjective ($|\mathscr{K}| \geq |\mathscr{C}|$).
2. $\mathscr{E}$ is free iff for all $p \in \mathscr{P}$ $\tilde{p}$ is injective ($|\mathscr{K}| \leq |\mathscr{C}|$).
3. $\mathscr{E}$ is regular iff for all $p \in \mathscr{P}$ $\tilde{p}$ is bijective ($|\mathscr{K}| = |\mathscr{C}|$).
4. $|\mathscr{P}| = |\mathscr{C}|$ iff $\mathscr{E}_e : \mathscr{P} \to \mathscr{C}$ is bijective for one $e \in \mathscr{K}$.
5. If $\mathscr{E}$ is free then $|\mathscr{K}| = |\mathscr{C}|$ iff for all $p \in \mathscr{P}$ $\tilde{p}$ is bijective.

□

Proof. Statements (1) – (3) follow trivially from Definition 4.12. Statement (4) follows from the injectivity of the maps $\mathscr{E}_e : \mathscr{P} \to \mathscr{C}$ due to the definition of Π. Statement (5) follows from the injectivity condition of statement (2), ■

Lemma 4.14. If a cryptosystem Π is perfectly secret then Π is transitive. □

Proof. Assume the contrary, $\mathscr{E}$ is not transitive. Thus there exists $p \in \mathscr{P}$ such that $\tilde{p} : \mathscr{K} \to \mathscr{C}$ is not surjective. So we can choose a $c \in \mathscr{C} \setminus \tilde{p}(\mathscr{K})$. By construction $\mu_{\mathscr{P}|\mathscr{C}}(p|c) = 0$. By Convention 4.7 the map $\Omega \to \mathscr{C}$ is surjective, so there exists $(p', e) \in \Omega$ such that $\mathscr{E}_e(p') = c$. Now we choose $\mu_{\mathscr{P}}$ such that $\mu_{\mathscr{P}}(p), \mu_{\mathscr{P}}(p') > 0$. Since $\mu_{\mathscr{K}}(e) > 0$ it follows that $\mu_{\mathscr{C}}(c) \geq \mu_{\mathscr{P}}(p')\mu_{\mathscr{K}}(e) > 0$. We have $\mu_{\mathscr{C}}(c) > 0$ and $\mu_{\mathscr{P}|\mathscr{C}}(p|c) = 0 \neq \mu_{\mathscr{P}}(p)$, thus Π is not perfectly secret. ■

Now we can easily conclude:

Corollary 4.15. Let a cryptosystem Π be perfectly secret and free. Then Π is regular and $|\mathscr{K}| = |\mathscr{C}|$. □

Proof. Lemma 4.14 together with Lemma 4.13. ■

Example 4.16. Easy examples of regular cryptosystems can be constructed in the following way:

1. Let $(G, \circ)$ be a finite group and set $\mathscr{P} = \mathscr{C} = \mathscr{K} = G$. Then we can define $\mathscr{E}_e(p) = e \circ p$ (or $\mathscr{E}_e(p) = p \circ e$). Due to the group structure $\mathscr{E}$ is regular.

2. Let $\mathscr{P} = \{p_1, p_2\}$, $\mathscr{K} = \{e_1, e_2, e_3, e_4\}$ and let $\mathscr{C} = \{c_1, c_2, c_3, c_4\}$. We define $\mathscr{E}$ via the encryption matrix

$\mathscr{E}$	p_1	p_2
e_1	c_1	c_2
e_2	c_2	c_1
e_3	c_3	c_4
e_4	c_4	c_3

By construction the maps $\tilde{p_1} : \mathscr{K} \to \mathscr{C}$, $e \mapsto \mathscr{E}_e(p_1)$ and $\tilde{p_2} : \mathscr{K} \to \mathscr{C}$, $e \mapsto \mathscr{E}_e(p_2)$ are bijectvie, thus Π is regular.

□

The probability distribution $\mu_{\mathscr{C}}$ depends on $\mu_{\mathscr{P}}$ resp. $\mu_{\mathscr{K}}$, whereas the following lemma makes this relation explicit.

Lemma 4.17. Let Π be a cryptosystem.

1. Let $|\mathscr{P}| = |\mathscr{C}|$. If $\mu_{\mathscr{P}}$ is uniformly distributed then $\mu_{\mathscr{C}}$ is uniformly distributed.

2. Let $\mathscr{E}$ be regular. If $\mu_{\mathscr{K}}$ is uniformly distributed then $\mu_{\mathscr{C}}$ is uniformly distributed.

□

Proof. We use Lemma 4.13.

1. Let $|\mathscr{P}| = |\mathscr{C}|$. $\mu_{\mathscr{P}}$ being uniformly distributed means that $\mu_{\mathscr{P}} = \frac{1}{|\mathscr{P}|}$ is constant for all $p \in \mathscr{P}$. This implies

$$\mu_{\mathscr{C}}(c) = \sum_{e\in\mathscr{K}} \mu\left(\mathscr{E}_e^{-1}(c), e\right) = \sum_{e\in\mathscr{K}} \mu_{\mathscr{P}}\left(\mathscr{E}_e^{-1}(c)\right)\mu_{\mathscr{K}}(e) = \frac{1}{|\mathscr{P}|}\sum_{e\in\mathscr{K}} \mu_{\mathscr{K}}(e) = \frac{1}{|\mathscr{P}|}\cdot 1 = \frac{1}{|\mathscr{C}|}.$$

Thus $\mu_{\mathscr{C}}$ is uniformly distributed.

2. $\mathscr{E}$ is regular, thus $\tilde{p}$ is bijective for all $p \in \mathscr{P}$ and $|\mathscr{K}| = |\mathscr{C}|$. Moreover, $\mu_{\mathscr{K}} = \frac{1}{|\mathscr{K}|}$. As above it holds that

$$\mu_{\mathscr{C}}(c) = \sum_{p\in\mathscr{P}} \mu\left(p, \tilde{p}^{-1}(c)\right) = \sum_{p\in\mathscr{P}} \mu_{\mathscr{P}}(p)\mu_{\mathscr{K}}\left(\tilde{p}^{-1}(c)\right) = \frac{1}{|\mathscr{K}|}\sum_{p\in\mathscr{P}} \mu_{\mathscr{P}}(p) = \frac{1}{|\mathscr{K}|}\cdot 1 = \frac{1}{|\mathscr{C}|}.$$

Thus $\mu_{\mathscr{C}}$ is uniformly distributed.

■

Remark 4.18. Note that until the end of this section we assume that $\mathscr{E}$ is free, that is $|\mathscr{K}| \leq |\mathscr{C}|$. Moreover it follows that transitivity is equivalent to regularity. □

We state three further lemmata we need in order to prove Shannon's theorem.

Lemma 4.19. Let Π be a cryptosystem, let $\mathscr{E}$ be regular and let $\mu_{\mathscr{P}}$ be arbitrary. Π is perfectly secret for $\mu_{\mathscr{P}}$ iff

$$\forall e \in \mathscr{K}, c \in \mathscr{C} : \mu_{\mathscr{K},\mathscr{C}}(e,c) = 0 \text{ or } \mu_{\mathscr{K}}(e) = \mu_{\mathscr{C}}(c).$$

□

Proof. For perfect secrecy for $\mu_{\mathscr{P}}$ we have to show that for all $p \in \mathscr{P}, c \in \mathscr{C}$ it holds that $\mu_{\mathscr{P}}(p) = 0$ or $\mu_{\mathscr{C}|\mathscr{P}}(c|p) = \mu_{\mathscr{C}}(c)$.

⇒ Let $\mu_{\mathscr{K},\mathscr{C}}(e,c) > 0$, then there exists $p \in \mathscr{P}$ such that $\mathscr{E}_e(p) = c$ and $\mu_{\mathscr{P}}(p) > 0$. Then p is unique since $\mathscr{E}_e$ is injective. Since $\mathscr{E}$ is free, e is uniquely determined by p and c. Due to the independence of $\mathscr{P}$ and $\mathscr{K}$ we follow:

$$\mu_{\mathscr{P}}(p)\mu_{\mathscr{K}}(e) = \mu_{\mathscr{P},\mathscr{K}}(p,e) = \mu_{\mathscr{P},\mathscr{C}}(p,c) = \mu_{\mathscr{C}|\mathscr{P}}(c|p)\mu_{\mathscr{P}}(p) = \mu_{\mathscr{C}}(c)\mu_{\mathscr{P}}(p). \tag{4.1}$$

Since $\mu_{\mathscr{P}}(p) > 0$ we conclude that $\mu_{\mathscr{K}}(e) = \mu_{\mathscr{C}}(c)$.

⇐ Let $c \in \mathscr{C}$ and $p \in \mathscr{P}$ such that $\mu_{\mathscr{P}}(p) > 0$. $\mathscr{E}$ being regular implies that there exists exactly one $e \in \mathscr{K}$ such that $\mathscr{E}_e(p) = c$. By convention $\mu_{\mathscr{K}}(e) > 0$ thus $\mu_{\mathscr{K},\mathscr{C}}(e,c) > 0$. Thus by assumption we have $\mu_{\mathscr{C}}(c) = \mu_{\mathscr{K}}(e)$. Using Equation 4.1 again, we get $\mu_{\mathscr{C}}(c) = \mu_{\mathscr{C}|\mathscr{P}}(c|p)$.

■

Lemma 4.20. Let Π be a cryptosystem and let $\mathscr{E}$ be regular. Then Π is perfectly secret if $\mu_{\mathscr{K}}$ is uniformly distributed. □

Proof. By Lemma 4.17 $\mu_{\mathscr{C}}$ is uniformly distributed. Since $|\mathscr{K}| = |\mathscr{C}|$ holds we get $\mu_{\mathscr{K}}(e) = \frac{1}{|\mathscr{K}|} = \frac{1}{|\mathscr{C}|} = \mu_{\mathscr{C}}(c)$. Now apply Lemma 4.19. ■

Lemma 4.21. Let Π be a cryptosystem, let $\mathscr{E}$ be regular and let $\mu_{\mathscr{P}}$ be arbitrary. If Π is perfectly secret for $\mu_{\mathscr{P}}$ and if $\mu_{\mathscr{C}}$ is uniformly distributed then $\mu_{\mathscr{K}}$ is uniformly distributed. □

Proof. Let $e \in \mathscr{K}$. Choose $p \in \mathscr{P}$ with $\mu_{\mathscr{P}}(p) > 0$. We set $c := \mathscr{E}_e(p)$ and get $\mu_{\mathscr{K},\mathscr{C}}(e,c) > 0$. Thus $\mu_{\mathscr{K}}(e) = \mu_{\mathscr{C}}(c)$ by Lemma 4.19. ■

Now we can state Shannon's theorem.

Theorem 4.22 (Shannon)**.** Let Π be regular and $|\mathscr{P}| = |\mathscr{C}|$. The following statements are equivalent:

1. Π is perfectly secret for $\overline{\mu_{\mathscr{P}}}$ (uniform distribution).

2. Π is perfectly secret.

3. $\mu_{\mathscr{K}}$ is uniformly distributed.

□

Proof.

(1) ⇒ (3) By Lemma 4.17 $\mu_{\mathscr{C}}$ is uniformly distributed. Since Π is regular it follows by Lemma 4.21 that $\mu_{\mathscr{K}}$ is uniformly distributed.

(3) ⇒ (2) Follows by Lemma 4.20.

(2) ⇒ (1) Trivial.

■

A well-known perfectly secret cryptosystem is the **Vernam One-Time-Pad** that was invented by Frank Miller in 1882 and patented by Gilbert S. Vernam in 1919.

Example 4.23 (Vernam One-Time-Pad)**.** Let $n \in \mathbb{N}_{>0}$. We set $\mathscr{P} = \mathscr{C} = \mathscr{K} = (\mathbb{Z}/2\mathbb{Z})^n$. The keys from $\mathscr{K}$ are chosen randomly and uniformly distributed. For each $e \in \mathscr{K}$ we define $\mathscr{E}_e(p) := p \oplus e \in \mathscr{C}$ and $\mathscr{D}_e(c) := c \oplus e \in \mathscr{P}$. By construction this cryptosystem is perfectly secret due to Theorem 4.22. □

Clearly, there is a problem in applying Vernam's One-Time-Pad in practice. Still there might be scenarios, for example, ambassadors or intelligence agencies, were one-time-pads are used.

Note that the condition of randomly choosing the keys is the other practical bottleneck as we will see later on.

4.3 Entropy

In information theory the term **entropy** defines the expected value or average of the information in each message of a transmission. In more detail, the amount of information of every event forms a random variable (see Section 4.1) whose expected value is the entropy. There are different units of entropy, here we use a bit (which is due to Shannon's definition of entropy). Thus we assume bit representations of information.

For us, the entropy will be the key property in order to get rid of the assumption $|\mathscr{P}| = |\mathscr{C}|$ in Shannon's theorem (4.22).

Definition 4.24. Let $X : \Omega \to M$ be a random variable for some finite set M. The **entropy** of X is defined by

$$H(X) := -\sum_{x \in M} \mu_X(x) \lg \mu_X(x)$$

where lg is shorthand notation for $\log_2$.[2] □

Convention 4.25. In the following we will often abuse notation for X. So X denotes the underlying set or language $M = X$ as well as the corresponding random variable X. □

Remark 4.26.

1. Since $\lim_{a \to 0} a \lg a = 0$ we define $0 \lg 0 = 0$.
2. From the properties of the logarithm we get also the alternative formulation

$$H(X) = \sum_{x \in M} \mu_X(x) \lg \frac{1}{\mu_X(x)}.$$

[2] What we call **entropy** is also known as **binary entropy**.

□

Lemma 4.27. Let $X : \Omega \to M$ be a random variable for some finite set M. It holds that $H(X) \geq 0$. In particular, $H(X) = 0$ iff $\mu_X(x) = 1$ for some $x \in M$. □

Proof. Since $\mu_X(x) \in [0,1]$ $-\mu_X(x) \lg \mu_X(x) \geq 0$ for all $x \in M$. Moreover, $-\mu_X(x) \lg \mu_X(x) = 0$ iff $\mu_X(x) = 0$ (see Remark 4.26) or $\mu_X(x) = 1$. ■

Example 4.28.

1. Let's assume we throw a coin with the events 0 and 1. Let $\mu_X(0) = \frac{3}{4}$ and let $\mu_X(1) = \frac{1}{4}$. Then
$$H(X) = \frac{3}{4} \lg \frac{4}{3} + \frac{1}{4} \lg 4 \approx 0.81.$$
If we choose $\mu_X(0) = \mu_X(1) = \frac{1}{2}$ we get
$$H(X) = \frac{1}{2} \lg 2 + \frac{1}{2} \lg 2 = 1.$$
2. Let $|X| =: n < \infty$. If X is uniformly distributed then
$$H(X) = \sum_{i=1}^{n} \frac{1}{n} \lg n = \lg n.$$

□

Important for structuring messages into blocks of information are encodings.

Definition 4.29. Let X be a random variable and let $f : X \to \{0,1\}^\bullet$ be a map.

1. f is called an **encoding** of X if the extension to $X^\bullet$
$$f : X^\bullet \to \{0,1\}^\bullet,\ x_1 \cdots x_n \mapsto f(x_1) \cdots f(x_n)$$
is an injective map for all $n \in \mathbb{N}_{>0}$.
2. f is called **prefix-free** if there do not exist two elements $x, y \in X$ and an element $z \in \{0,1\}^\bullet$ such that $f(x) = f(y)z$.
3. The **average length of** f is defined by
$$\ell(f) := \sum_{x \in M} \mu_X(x) \mathrm{sz}_2(f(x)).$$

□

Example 4.30.

1. Let $X = \{a, b, c, d\}$ and consider the following maps $f, g, h : X \to \{0,1\}^\bullet$:

$f(a) = 1$	$g(a) = 0$	$h(a) = 0$
$f(b) = 10$	$g(b) = 10$	$h(b) = 01$
$f(c) = 100$	$g(c) = 110$	$h(c) = 10$
$f(d) = 1000$	$g(d) = 111$	$h(d) = 11$

a) f is an encoding: Start at the end and read the encoding backwards. Once a 1 is reached the signal ends.

b) g is an encoding: One can start at the beginning and reads the encoding forwards. Every time a known substring is found, this part is cut off: Let 10101110 be an encoding via g. We can read it as 10 10 111 0 thus we see: $g(bbda) = 10\ 10\ 111\ 0$. Moreover, g is a prefix-free encoding.

c) h is not an encoding since it is not injective on the extension to $X^\bullet$: $h(ac) = 010 = h(ba)$.

2. A well-known encoding was patented in 1947 by Frank Gray, a researcher at Bell Labs. Nowadays people usually mean the **reflected binary code** when they are speaking about the **Gray code**. It is an encoding for handling error corrections and is used in digital communication like some cable TV systems.

   The main idea is to use a different binary representation of natural numbers. We already have the usual binary code, so what's the problem with it?

Decimal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

   When we think about digital transmission changes of state are handled by physical switches that are not ideal in the sense that they might not change state synchronously. If we are looking at the binary representation above, when changing the state from decimal 3 to decimal 4 three switches have to change at the same time: 011 ⇒ 100. During the period of changing all three states the system might read false information, like 011 ⇒ 010 ⇒ 110 ⇒ 100. In order to omit these intermediate, asynchronous states Gray's idea was to find a binary presentation that only changes one switch at a time.

   The nice fact is that Gray's code can be iteratively constructed:

Decimal	Gray
0	0
1	1

Decimal	Gray
0	00
1	01
2	11
3	10

Decimal	Binary
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

We can see that in Gray's code going from one state to the next we only change exactly one switch. Moreover, the code is even reflected on the center line presented in the above tables: Generating a 3-digit Gray code, we use the 2-digit Gray code. We reflect the 2-digit code on the centered line. Now for the upper half we add 0 as first digit, for the lower half we add 1 as first digit.

Moreover, this Gray code is even **cyclic**: Cyclic means that once we reach the end (7 resp. 100) we can start back with the beginning (0 resp. 000) and again change only one switch.

□

The idea is now that the entropy of X should be $\ell(f)$ if f is the most efficient encoding for X. Here "most efficient" means that an event with probability $0 < a < 1$ is encoded by $f(a)$ such that $\mathrm{sz}_2\left((f(a))\right) = -\lg a = \lg \frac{1}{a}$.

In the following example we descibe **Huffman's algorithm** that produces exactly such an encoding.

Example 4.31. Let $M = \{a, b, c, d, e\}$ and let X be an M-valued random variable with probability distribution $\mu_X(a) = 0.05$, $\mu_X(b) = 0.10$,$\mu_X(c) = 0.12$, $\mu_X(d) = 0.13$ and $\mu_X(e) = 0.60$, Huffman's algorithm works as follows: Consider the elements of X as vertices in a graph. Take the two elements x, y of X with lowest probabilities $\mu_X(x)$ and $\mu_X(y)$. Now connect x and y to a new vertex z with probability $\mu_X(z) = \mu_X(x) + \mu_X(y)$ and label the two directed edges by 0 resp. 1. Now forget x and y and start the process again. The algorithm terminates once the vertex with probability 1 is introduced in the graph.

We can illustrate this as shown in Figure 4.1.

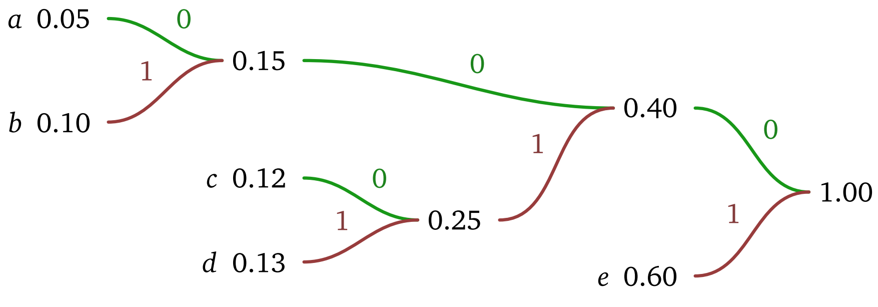


Figure 4.1: Steps of Huffman's algorithm

Thus we generated a prefix-free encoding f via

x	$f(x)$
a	000
b	001
c	010
d	011
e	1

The average length of f is

$$\ell(f) = 0.05 \cdot 3 + 0.10 \cdot 3 + 0.12 \cdot 3 + 0.13 \cdot 3 + 0.6 \cdot 1 = 1.8$$

whereas the entropy is $H(X) = \sum_{x \in M} \mu_X(x) \lg \mu_X(x) \approx 1.75$. □

It follows that we can relate $\ell(f)$ and $H(X)$ to each other.

Theorem 4.32. There exists an encoding $f : X \to \{0,1\}^\bullet$ such that

$$H(X) \leq \ell(f) \leq H(X) + 1.$$

□

Proof. Huffman's algorithm from Example 4.31 produces such an encoding f. ■

Definition 4.33. Let Σ be an alphabet.

1. If X is a random variable with $X \subset \Sigma^m$ for $m \in \mathbb{N}_{>0}$ such that $n = |\Sigma^m|$. Then we define the **redundancy of** X by
$$R(X) := \lg n - H(X).$$
Since $0 \leq H(X) \leq \lg n$ it also holds that $0 \leq R(X) \leq \lg n$. Moreover: $R(X) + H(X) = \lg n$.
2. Let X_n be a random variable for $X_n \subset \Sigma^n$ of n-digit words in a language $X \subset \Sigma^\bullet$.
   a) The **entropy of** X **(per letter)** is defined as
   $$H_X := \lim_{n \to \infty} \frac{H(X_n)}{n}.$$
   b) The **redundancy of** X **(per letter)** is defined as
   $$R_X := \lg |\Sigma| - H_X = \lim_{n \to \infty} \frac{R(X_n)}{n}.$$

□

Example 4.34. Let X be an M-valued random variable for $X = M = \Sigma = \{a, \ldots, z\}$. If we assume μ_X to be uniformly distributed then $H(X) = \lg 26 \approx 4.70$ (so we need between 5 and 6 bits). If we take μ_X as the distribution of the English language then $H(X) \approx 4.19$. Next we get $H(X_1) = H(X)$ and $H(X_2) \approx 3.19$. Empirical data shows that

$$1.0 \leq H_X \leq 1.5.$$

If we assume $H_X = 1.25 \approx 0.27 \cdot \lg |\Sigma|$ the redundancy $R_X = \lg |\Sigma| - H_X = 4.70 - 1.25 = 3.45 \approx 0.73 \cdot \lg |\Sigma|$.

What does this mean? Assume that t_n is the number of equally probable texts in English language for text beginning of n letters. By $H_X = \lim_{n\to\infty} \frac{\lg t_n}{n} \approx 1.25$ we get $t_n \approx 2^{1.25\cdot n}$ for large n. For example, taking $n = 20$ we get $t_{20} \approx 3.35 \cdot 10^7$ whereas $|\Sigma^{20}| = 26^{20} \approx 1.99 \cdot 10^{28}$. So if we want to attack some cryptosystem based on the English language and its probability distribution, a brute force attack can be optimized quite a lot, for example, by narrowing the plaintext space of 20 letter words. □

Remark 4.35. Note that a single text does not have an entropy. Only languages have an entropy. □

We need some more ingredients in order to apply our knowledge of entropy and redundancy to cryptosystems.

Definition 4.36. Let $X : \Omega \to \mathcal{X}$ and $Y : \Omega \to \mathcal{Y}$ be two random variables with $|\mathcal{X}|, |\mathcal{Y}| < \infty$.

1. The **joint entropy of** X **and** Y is defined by
$$H(X,Y) := -\sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \mu_{X,Y}(x,y) \lg \mu_{X,Y}(x,y).$$

2. The **conditional entropy or equivocation of** X **and** Y is defined by
$$H(X|Y) := \sum_{y\in\mathcal{Y}} \mu_Y(y) H(X|y)$$
where
$$H(X|y) := -\sum_{x\in\mathcal{X}} \mu_{X|Y}(x|y) \lg \mu_{X|Y}(x|y).$$

3. The **transinformation of** X **and** Y is defined by
$$I(X,Y) := H(X) - H(X|Y).$$

□

Theorem 4.37. Let $X : \Omega \to \mathcal{X}$ and $Y : \Omega \to \mathcal{Y}$ be two random variables with $|\mathcal{X}|, |\mathcal{Y}| < \infty$.

1. $H(X) \leq \lg |\mathcal{X}|$. Equality holds iff μ_X is uniformly distributed.
2. $H(X|Y) \leq H(X)$. Equality holds iff X and Y are independent.
3. $H(X|Y) = H(X,Y) - H(Y)$.
4. $H(X,Y) \leq H(X) + H(Y)$. Equality holds iff X and Y are independent.
5. $H(X|Y) = H(Y|X) + H(X) - H(Y)$.
6. $I(X,Y) = I(Y,X) \geq 0$.

□

Proof.

1. Exercise.

2. This is a bit harder to prove, see, for example, Theorem 1.17 in [Mos17].
3. Exercise.
4. Exercise, where the equality statement follows from (2) and (3).
5. Follows from (3) taking into account that $H(X,Y) = H(Y,X)$.
6. The equality follows from (5), the non-negativeness follows from (2). ■

Remark 4.38. Let X be a random variable and denote by X^n the random variable describing the n-fold independent repetition of the event X. Then $H(X^n) := H(X,\ldots,X) = nH(X)$. □

Convention 4.39. Until the end of this chapter we assume that Π is a symmetric cryptosystem such that $\mathscr{P},\mathscr{C},\mathscr{K}$ are finite ($|\mathscr{C}| \geq |\mathscr{P}|$ since $\mathscr{E}_e$ is injective), for all $e \in \mathscr{K}$ $\mathscr{E}_e$ is a map, and $\mathscr{P}$ and $\mathscr{K}$ are independent. □

Definition 4.40. In the above setting we define

1. the **key equivocation** $H(\mathscr{K}|\mathscr{C})$ resp. the **plaintext equivocation** $H(\mathscr{P}|\mathscr{C})$, and
2. the **key transinformation** $I(\mathscr{K},\mathscr{C})$ resp. the **plaintext transinformation** $I(\mathscr{P},\mathscr{C})$.

□

Lemma 4.41. Let Π be a cryptosystem as in Convention 4.39.

1. $H(\mathscr{P},\mathscr{K}) = H(\mathscr{K},\mathscr{C}) = H(\mathscr{P},\mathscr{C},\mathscr{K})$.
2. $H(\mathscr{C}) \geq H(\mathscr{C}|\mathscr{K}) = H(\mathscr{P}|\mathscr{K}) = H(\mathscr{P})$.
3. $H(\mathscr{K}|\mathscr{C}) = H(\mathscr{P}) + H(\mathscr{K}) - H(\mathscr{C})$.
4. $I(\mathscr{K},\mathscr{C}) = H(\mathscr{C}) - H(\mathscr{P}) \geq 0$.

□

Proof.

1. $H(\mathscr{P},\mathscr{C},\mathscr{K}) = H(\mathscr{C},\mathscr{P},\mathscr{K}) = H(\mathscr{C},(\mathscr{P},\mathscr{K})) = H(\mathscr{C}|(\mathscr{P},\mathscr{K})) + H(\mathscr{P},\mathscr{K})$. By our assumptions $H(\mathscr{C}|(\mathscr{P},\mathscr{K})) = 0$ as $\mathscr{C}$ is defined by $\mathscr{P}$ and $\mathscr{K}$: Since $\mathscr{E}_e$ is injective for all $e \in \mathscr{K}$ we have $c = \mathscr{E}_e(p)$ and thus $p = \mathscr{D}_e(c)$. It follows that $H(\mathscr{P},\mathscr{C},\mathscr{K}) = H(\mathscr{P},\mathscr{K}) = H(\mathscr{K},\mathscr{C})$.
2. $H(\mathscr{C}) \geq H(\mathscr{C}|\mathscr{K})$ is clear. For the equation we use (1):
$$H(\mathscr{C}|\mathscr{K}) = H(\mathscr{C},\mathscr{K}) - H(\mathscr{K}) = H(\mathscr{P},\mathscr{K}) - H(\mathscr{K}) = H(\mathscr{P}) + H(\mathscr{K}) - H(\mathscr{K}) = H(\mathscr{P}).$$
3. Again we use (1):
$$H(\mathscr{K}|\mathscr{C}) = H(\mathscr{K},\mathscr{C}) - H(\mathscr{C}) = H(\mathscr{P},\mathscr{K}) - H(\mathscr{C}) = H(\mathscr{P}) + H(\mathscr{K}) - H(\mathscr{C}).$$
4. We use (3):
$$I(\mathscr{K},\mathscr{C}) = H(\mathscr{K}) - H(\mathscr{K}|\mathscr{C}) = H(\mathscr{K}) - H(\mathscr{P}) - H(\mathscr{K}) + H(\mathscr{C}) = H(\mathscr{C}) - H(\mathscr{P}).$$
Now by (2) it follows that $H(\mathscr{C}) - H(\mathscr{P}) \geq 0$. ■

Remark 4.42. Note that $H(\mathscr{P}) \leq H(\mathscr{C})$ generalizes Lemma 4.17: If $|\mathscr{P}| = |\mathscr{C}|$ and $\mu_{\mathscr{P}}$ is uniformly distributed, then $\mu_{\mathscr{C}}$ is also uniformly distributed. Moreover, $H(\mathscr{P}) < H(\mathscr{C})$ is possible, for example, if Π is perfectly secret, $|\mathscr{P}| = |\mathscr{C}|$ and $\mathscr{P}$ is not uniformly distributed. □

Definition 4.43. In the above setting we denote by $R(\mathscr{P}) := \lg|\mathscr{P}| - H(\mathscr{P})$ the **redundancy of** $\mathscr{P}$. □

Lemma 4.44. Let Π be a cryptosystem and let $|\mathscr{P}| = |\mathscr{C}|$. Then

$$H(\mathscr{K}) \geq H(\mathscr{K}|\mathscr{C}) \geq H(\mathscr{K}) - R(\mathscr{P})$$

and

$$R(\mathscr{P}) \geq I(\mathscr{K}, \mathscr{C}) \geq 0.$$

□

Proof. Using Lemma 4.41 (3) and $H(\mathscr{C}) \leq \lg|\mathscr{C}| = \lg|\mathscr{P}|$ we get that

$$H(\mathscr{K}|\mathscr{C}) \geq H(\mathscr{K}) + H(\mathscr{P}) - \lg|\mathscr{P}| = H(\mathscr{K}) - R(\mathscr{P}).$$

In the same way we conclude with Lemma 4.41 (4) that

$$0 \leq I(\mathscr{K}, \mathscr{C}) = H(\mathscr{C}) - H(\mathscr{P}) \leq \lg|\mathscr{C}| - H(\mathscr{P}) = \lg|\mathscr{P}| - H(\mathscr{P}) = R(\mathscr{P}).$$

■

Example 4.45. Let $\mathscr{P} = \{a, b\}, \mathscr{C} = \{c, d\}, \mathscr{K} = \{e_1, e_2\}$ with encryption matrix

$\mathscr{E}$	a	b
e_1	c	d
e_2	d	c

Moreover, we choose $\mu_{\mathscr{P}}(a) = \frac{1}{4}, \mu_{\mathscr{P}}(b) = \frac{3}{4}$, $\mu_{\mathscr{K}}(e_1) = \frac{1}{4}$, and $\mu_{\mathscr{K}}(e_2) = \frac{3}{4}$. Then we get $\mu_{\mathscr{C}}(c) = \frac{10}{16}$ and $\mu_{\mathscr{C}}(d) = \frac{6}{16}$. It follows that $H(\mathscr{P}) = H(\mathscr{K}) \approx 0.81$ and $H(\mathscr{C}) \approx 0.95$. We get $R(\mathscr{P}) = 1 - H(\mathscr{P}) \approx 0.19$ and $H(\mathscr{K}) - R(\mathscr{P}) \approx 0.62$. Thus

$$0.62 \leq H(\mathscr{K}|\mathscr{C}) \leq 0.81$$

and

$$0 \leq I(\mathscr{K}, \mathscr{C}) \leq 0.19.$$

This fits with the direct computations

$$\begin{array}{rcll} H(\mathscr{K}|\mathscr{C}) & = & H(\mathscr{P}) + H(\mathscr{K}) - H(\mathscr{C}) & \approx & 0.67, \\ I(\mathscr{K}, \mathscr{C}) & = & H(\mathscr{C}) - H(\mathscr{P}) & \approx & 0.14. \end{array}$$

□

Remark 4.46. The last lemma states the following important relations:

1. $R(\mathscr{P})$ is a (mostly good) upper bound for the key transinformation.

2. We need at least as much key entropy $H(\mathscr{K})$ as we have redundancy in $\mathscr{P}$ in order to get a nonnegative lower bound for the key equivocation $H(\mathscr{K}|\mathscr{C}) \geq H(\mathscr{K}) - R(\mathscr{P})$.
3. If $\mathscr{P}$ is uniformly distributed then $R(\mathscr{P}) = 0$ thus also $I(\mathscr{K}, \mathscr{C}) = 0$ and $H(\mathscr{K}) = H(\mathscr{K}|\mathscr{C})$.

□

Example 4.47. Let $\mathscr{K} = \mathscr{C} = \Sigma^k$ and let $L \subset \Sigma^\bullet$ be a language with entropy H_L and redundancy R_L.[3] Let $\mathscr{P} \subset L$. For n big enough we get

$$H(\mathscr{K}|\mathscr{C}) \geq H(\mathscr{K}) - R(\mathscr{P}) \approx H(\mathscr{K}) - nR_L.$$

In other words: If $H(\mathscr{K})$ is fixed and n can grow, for example via repeated encryption with the same key, then the entropy is exhausted as n increases. □

This leads to the following definition:

Definition 4.48. With notation as in Example 4.47 the number $n_0 := \left\lceil \frac{H(\mathscr{K})}{R_L} \right\rceil$ is called the **unicity distance**. □

Remark 4.49. The higher the redundancy of a language the faster a key is exhausted. So one needs to compensate the redundancy of a language, for example, by increasing the key size. □

Let us look on values for n_0 in practical examples:

Example 4.50. Let $|\Sigma| = 26$ and L be the English language with $R_L = 3.45$.

symmetric cryptosystem	$\|\mathscr{K}\|$	$H(\mathscr{K})$	n_0
monoalphabetic substitution	$26! \approx 2^{88.4}$	≈ 88.4	26
permutation of 16-blocks	$16! \approx 2^{44.3}$	≈ 44.3	13
"DES" (see Section 6.2)	2^{56}	56	17
"AES" (see Section 6.3)	2^{128}	128	38

Let us assume a text length of $n = 20$ and the monoalphabetic substitution. Then we get

$$H(\mathscr{K}|\mathscr{C}) \geq H(\mathscr{K}) - R(\mathscr{P}) = H(\mathscr{K}) - 20R_L \approx 88.40 - 20 \cdot 3.45 = 19.40.$$

Thus we have approximately $2^{19.4} \approx 691\,802$ keys to try. This is done in seconds on a usual personal computer. □

So what we want to do is to increase the unicity distance but still keep the key lengths short in order to have a fast and efficient cryptosystem. There are several attempts to do this:

Remark 4.51.

1. Reduce the redundancy of $\mathscr{P}$, for example, by compressing (zipping) the text.

[3] Note that one can define the entropy of a language with non fixed word length in more generality, also we do not do this here.

2. Cover the redundancy of $\mathscr{P}$ against attackers with limited computing power, for example, by using combinations of substitution and so-called **Feistel ciphers** (see Section 6.1)
3. Try to bloat the key entropy against attackers with limited computing power by using so-called **autokey ciphers**: These ciphers include the plaintext in the key itself, for example, when using Vigenère's cipher with autokey we would have something like this:

plaintext	`dies`	`erte`	`xtxx`
key	`keyd`	`iese`	`rtex`

4. Use pseudo random sequences for the key. Still, getting pseudo random elements is not trivial at all.

□

We finish this chapter with further investigations on free cryptosystems.

Lemma 4.52. Let Π be a cryptosystem.

1. $H(\mathscr{P},\mathscr{K}) = H(\mathscr{P},\mathscr{C}) + H(\mathscr{K}|(\mathscr{P},\mathscr{C}))$.
2. $H(\mathscr{K}) = H(\mathscr{C}|\mathscr{P}) + H(\mathscr{K}|\mathscr{P},\mathscr{C})$.
3. $H(\mathscr{K}|\mathscr{C}) = H(\mathscr{P}|\mathscr{C}) + H(\mathscr{K}|\mathscr{P},\mathscr{C})$.
4. Π free $\Longleftrightarrow H(\mathscr{K}|(\mathscr{P},\mathscr{C})) = 0 \Longleftrightarrow I(\mathscr{K},(\mathscr{P},\mathscr{C})) = H(\mathscr{K})$.

□

Proof.

1. $H(\mathscr{K}|(\mathscr{P},\mathscr{C})) = H(\mathscr{K},\mathscr{P},\mathscr{C}) - H(\mathscr{P},\mathscr{C}) = H(\mathscr{K},\mathscr{P}) - H(\mathscr{P},\mathscr{C})$.
2. By assumption $\mathscr{P}$ and $\mathscr{K}$ are independent, thus $H(\mathscr{P}|\mathscr{K}) = H(\mathscr{P})$.

$$\begin{aligned} H(\mathscr{P}|\mathscr{K}) = H(\mathscr{P},\mathscr{K}) - H(\mathscr{K}) &\overset{(1)}{=} H(\mathscr{P},\mathscr{C}) + H(\mathscr{K}|(\mathscr{P},\mathscr{C})) - H(\mathscr{K}) \\ &= H(\mathscr{C}|\mathscr{P}) + H(\mathscr{P}) + H(\mathscr{K}|(\mathscr{P},\mathscr{C})) - H(\mathscr{K}). \end{aligned}$$

It follows:

$$H(\mathscr{K}) = H(\mathscr{C}|\mathscr{P}) + H(\mathscr{P}) - H(\mathscr{P}|\mathscr{K}) + H(\mathscr{K}|(\mathscr{P},\mathscr{C})).$$

By assumption $H(\mathscr{P}) - H(\mathscr{P}|\mathscr{K}) = 0$ thus the statement follows.

3. Using (1) we get

$$\begin{aligned} H(\mathscr{K}|\mathscr{C}) &= H(\mathscr{K},\mathscr{C}) - H(\mathscr{C}) = H(\mathscr{P},\mathscr{K}) - H(\mathscr{C}) \\ &= H(\mathscr{P},\mathscr{C}) + H(\mathscr{K}|(\mathscr{P},\mathscr{C})) - H(\mathscr{C}) = H(\mathscr{P}|\mathscr{C}) + H(\mathscr{K}|(\mathscr{P},\mathscr{C})). \end{aligned}$$

4. $I(\mathscr{K},(\mathscr{P},\mathscr{C})) = H(\mathscr{K}) - H(\mathscr{K}|(\mathscr{P},\mathscr{C}))$. If Π is free then the map $\tilde{p} : \mathscr{K} \to \mathscr{C},\ e \mapsto \mathscr{E}_e(p)$ is injective for all $p \in \mathscr{P}$. So $\mu_{\mathscr{K}|(\mathscr{P},\mathscr{C})}(e|(p,c))$ is 0 if the combination is not possible and 1 if $\mathscr{E}_e(p) = c$ and thus $H(\mathscr{K}|\mathscr{P},\mathscr{C}) = 0$. If $H(\mathscr{K}|(\mathscr{P},\mathscr{C})) = 0$ then each $e \in \mathscr{K}$ is uniquely defined by p and c, thus the map $\tilde{p}$ is injective for all $p \in \mathscr{P}$. So Π is free. ■

Remark 4.53. We can interpret $H(\mathscr{K}|(\mathscr{P},\mathscr{C}))$ as the **unused key entropy** and $I(\mathscr{K}|(\mathscr{P},\mathscr{C}))$ as the **used key entropy**. □

Theorem 4.54. Let Π be a free cryptosystem.

1. $H(\mathscr{P}|\mathscr{C}) = H(\mathscr{K}|\mathscr{C}) = H(\mathscr{P}) + H(\mathscr{K}) - H(\mathscr{C})$.
2. $I(\mathscr{P},\mathscr{C}) = H(\mathscr{C}) - H(\mathscr{K})$.
3. $H(\mathscr{K}) \leq H(\mathscr{C})$.

□

Proof. By Lemma 4.52 (3) we have $H(\mathscr{P}|\mathscr{C}) = H(\mathscr{K}|\mathscr{C}) - H(\mathscr{K}|\mathscr{P},\mathscr{C})$. Since Π is free it follows from 4.52 (4) that $H(\mathscr{K}|\mathscr{P},\mathscr{C}) = 0$. By Lemma 4.41 (3) we know that $H(\mathscr{K}|\mathscr{C}) = H(\mathscr{P}) + H(\mathscr{K}) - H(\mathscr{C})$ which proves statement (1). For statements (2) and (3) we see that

$$0 \leq I(\mathscr{P},\mathscr{C}) \overset{\text{by def.}}{=} H(\mathscr{P}) - H(\mathscr{P}|\mathscr{C}) \overset{(1)}{=} H(\mathscr{P}) - H(\mathscr{K}|\mathscr{C}) \overset{(1)}{=} H(\mathscr{C}) - H(\mathscr{K}).$$

■

Corollary 4.55. Let Π be a free cryptosystem.

1. If $|\mathscr{P}| = |\mathscr{C}|$ then $H(\mathscr{P}|\mathscr{C}) \geq H(\mathscr{K}) - R(\mathscr{P})$.
2. If $|\mathscr{K}| = |\mathscr{C}|$ then $I(\mathscr{P},\mathscr{C}) \leq R(\mathscr{K})$.

□

Proof.

1. By Theorem 4.54 (1) $H(\mathscr{P}|\mathscr{C}) = H(\mathscr{K}|\mathscr{C})$. The statement is then just Lemma 4.44.
2. By Theorem 4.54 (2) we have $I(\mathscr{P},\mathscr{C}) = H(\mathscr{C}) - H(\mathscr{K})$, thus
$$I(\mathscr{P},\mathscr{C}) = H(\mathscr{C}) - H(\mathscr{K}) \leq \lg|\mathscr{C}| - H(\mathscr{K}) = \lg|\mathscr{K}| - H(\mathscr{K}) = R(\mathscr{K}).$$

■

As already explained at the beginning of this section our goal is to generalize Shannon's theorem (4.22) by removing the assumption that $|\mathscr{P}| = |\mathscr{C}|$.

Theorem 4.56. Let Π be a cryptosystem as in Convention 4.39. Then the following statements are equivalent:

1. Π is perfectly secret for $\mu_{\mathscr{P}}$ (i.e. $\mathscr{P}$ and $\mathscr{C}$ are independent).
2. $I(\mathscr{P},\mathscr{C}) = 0$.
3. $H(\mathscr{P},\mathscr{C}) = H(\mathscr{P}) + H(\mathscr{C})$.
4. $H(\mathscr{C}|\mathscr{P}) = H(\mathscr{C})$.
5. $H(\mathscr{P}|\mathscr{C}) = H(\mathscr{P})$.

If we further assume that Π is free, and thus in our setting regular, the list of equivalences also includes:

6. $H(\mathscr{K}) = H(\mathscr{C})$.
7. $H(\mathscr{K}|\mathscr{C}) = H(\mathscr{P})$ which implies $H(\mathscr{K}) \geq H(\mathscr{P})$.

□

Proof. The equivalence of $(1) - (5)$ is trivial. By Lemma 4.52 (2) and (4) we know that $H(\mathscr{K}) = H(\mathscr{C}|\mathscr{P})$. So (6) follows by (4) and vice versa. By Lemma 4.52 (3) and (4) we have that $H(\mathscr{K}|\mathscr{C}) = H(\mathscr{P}|\mathscr{C})$. Thus (7) follows by (5) and vice versa. ■

Corollary 4.57 (Shannon)**.** Let Π be a free cryptosystem. Then Π is perfectly secret for $\mu_{\mathscr{P}}$ and $\mu_{\mathscr{C}}$ is uniformly distributed iff $|\mathscr{K}| = |\mathscr{C}|$ and $\mu_{\mathscr{K}}$ is uniformly distributed. □

Proof.

⇒ Since $\mu_{\mathscr{C}}$ is uniformly distributed we know that $H(\mathscr{C}) = \lg|\mathscr{C}|$. Π is free, so $|\mathscr{K}| \leq |\mathscr{C}|$. Since we assume perfect secrecy for $\mu_{\mathscr{P}}$ and Π is free we get $H(\mathscr{K}) = H(\mathscr{C}) = \lg|\mathscr{C}|$ by Theorem 4.56 (6). By definition $\mu_{\mathscr{K}}$ is then uniformly distributed and thus also $|\mathscr{K}| = |\mathscr{C}|$.

⇐ By Theorem 4.54(3) $H(\mathscr{K}) \leq H(\mathscr{C})$. Since $\mu_{\mathscr{K}}$ is uniformly distributed we have the inequality

$$\lg|\mathscr{K}| = H(\mathscr{K}) \leq H(\mathscr{C}) \leq \lg|\mathscr{C}|. \tag{4.2}$$

Since $|\mathscr{K}| = |\mathscr{C}|$ equality must hold in 4.2. Hence $\mu_{\mathscr{C}}$ is uniformly distributed, $H(\mathscr{K}) = H(\mathscr{C})$ and Π is perfectly secret for $\mu_{\mathscr{P}}$ by Theorem 4.56 $(1) \Longleftrightarrow (6)$. ■

Chapter 5

Pseudorandom Sequences

In the last chapters we have often spoken about randomness and pseudorandomness. Most of our assumptions on perfect secrecy and big enough unicity distances depend on random or pseudorandom input data. Next we try to give these notions some mathematical meaning.

5.1 Introduction

Possible sources of *random sequences* (RS) might be

- white noise captured by your laptops microphone,
- throwing a perfect coin, or
- quantum mechanical systems producing statistical randomness.

Pseudorandomness on the other hand starts with a short random source produced in a way like explained above. A pseudorandom sequence is then generated in a deterministic way, i.e. by an algorithm, having this *random seed* as input. *Pseudorandom sequences* (PRS) have advantages and disadvantages:

Advantages

1. PRS can be produced by software instead of hardware.
2. PRS can be reconstructed: Assume its usage in a cryptosystem. One just has to exchange the random seed and the algorithm generating the PRS.

Disadvantages

1. The seed must be random, otherwise the PRS can be easily recovered.
2. Moreover, the seed must be secret, otherwise the complete PRS is known.
3. Clearly, the algorithm producing the PRS is deterministic and can thus be attacked.

Possible applications of PRS are the generation of session keys (web session, messenger, etc.), stream ciphers (see Section 3.3) or the generation of TANs and PINs (online banking, etc.).

Example 5.1. One example we have already seen are linear feedback shift registers, see Definition 3.20. □

5.2 Linear recurrence equations and pseudorandom bit generators

Convention 5.2. In the following, let K be a field, let $\ell \in \mathbb{N}$ and let $c = (c_0, \ldots, c_{\ell-1})^{\mathrm{tr}} \in K^\ell$ with $c_0 \neq 0$. □

Definition 5.3. A **linear recurrence equation LRE for** c **of degree** $\ell > 0$ is given by

$$s_{n+\ell} = \begin{pmatrix} s_n & \cdots & s_{n+\ell-1} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ \vdots \\ c_{\ell-1} \end{pmatrix} \quad (n \geq 0) \tag{5.1}$$

such that

1. $t^{(0)} := t = \begin{pmatrix} t_0 & \cdots & t_{\ell-1} \end{pmatrix} \in K^{1\times\ell}$ is the *initial value*,
2. $s_i = t_i$ for $0 \leq i \leq \ell - 1$, and
3. $t^{(n)} := \begin{pmatrix} s_n & \cdots & s_{n+\ell-1} \end{pmatrix}$ is the *n-th state vector*.

Since the full information of the sequence s lies in c and t we also write $s := \langle c, t \rangle$. □

Example 5.4. Taking $K = \mathbb{F}_2$, $\ell = 4$, $c = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ and $t = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}$ we get:

$$s = \underline{1, 0, 1, 0,} 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, \mid \underline{1, 0, 1, 0,} \ldots$$

□

Remark 5.5. A linear recurrence equation over $K = \mathbb{F}_2$ (abbreviated by $\mathbb{F}_2$-LRE) of degree ℓ is nothing but an **ℓ-bit Linear Feedback Shift Register (LFSR)** [Wik17e]. It is an example of a *linear* **pseudorandom bit generator (PRBG)** (cf. Example 5.65). It is one among many pseudorandom bit generators [Wik17g]. □

Definition 5.6.

1. Define $\chi := \chi_c := x^\ell - c_{\ell-1}x^{\ell-1} - \cdots - c_1 x - c_0 \in K[x]$.
2. s is called k-periodic ($k \in \mathbb{N}$) if $s_{i+k} = s_i$ for all $i \geq 0$, or equivalently, $t^{(i+k)} = t^{(i)}$ for all $i \geq 0$.
3. c is called k-periodic ($k \in \mathbb{N}$) if $s = \langle c, t \rangle$ is k-periodic for all $t \in K^{1\times\ell}$.
4. If s (resp. c) is k-periodic for some $k \in \mathbb{N}$ then denote by $\operatorname{per}(s)$ (resp. $\operatorname{per}(c)$) the smallest such number and call it the **period length**. If such a k does not exist then set $\operatorname{per} s := \infty$ (resp. $\operatorname{per} c := \infty$).

□

Lemma 5.7. With s, c and t defined as in Definition 5.3 the following statements hold:

1. $s_{n+\ell} = t^{(n)} \cdot c$.

2. $t^{(n)} = t^{(n-1)} \cdot C = t \cdot C^n$ with

$$C := \begin{pmatrix} 0 & \cdots & \cdots & 0 & c_0 \\ 1 & 0 & \cdots & 0 & c_1 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & c_{\ell-2} \\ 0 & \cdots & 0 & 1 & c_{\ell-1} \end{pmatrix}.$$

3. $\langle c, t\rangle$ is k-periodic if and only if $t \cdot C^k = t$.
4. c is k-periodic if and only if $C^k = I_\ell$.
5. $\operatorname{per}\langle c, t\rangle = \min\{k > 0 \mid t \cdot C^k = t\}$.
6. $\operatorname{per} c = \min\{k > 0 \mid C^k = I_\ell\}$.
7. $\langle c, t\rangle$ is k-periodic iff $\operatorname{per}\langle c, t\rangle \mid k$.
8. $\operatorname{per} c = \operatorname{lcm}\{\operatorname{per}\langle c, t\rangle \mid t \in K^{1\times\ell}\}$.
9. $\operatorname{per}\langle c, 0\rangle = 1$.
10. C is the companion matrix[1] of χ. Hence, χ is the minimal polynomial and therefore also the characteristic polynomial of C (as its degree is ℓ).
11. C is a regular matrix since $c_0 \neq 0$, i.e., $C \in \mathrm{GL}_\ell(K)$.
12. $\operatorname{per} c = \operatorname{ord} C$ in $\mathrm{GL}_\ell(K)$.
13. The cyclic subgroup $\langle C\rangle$ generated by the matrix C acts on the vector space $K^{1\times\ell}$. The **orbit**[2] $t \cdot \langle C\rangle = \{t^{(i)} \mid i \in \mathbb{Z}\}$ of t is nothing but the set of all reachable state vectors.
14. $\operatorname{per}\langle c, t\rangle = |t \cdot \langle C\rangle|$.

□

Proof. (1) – (13) are trivial. To see (14) note that the state vectors $t^{(i)}$ with $0 \leq i < \operatorname{per}\langle c, t\rangle$ are pairwise distinct: $t^{(i)} = t^{(j)}$ for $0 \leq i \leq j < \operatorname{per}\langle c, t\rangle$ means that $tC^i = tC^j$ and hence $tC^{j-i} = t$ with $j - i < \operatorname{per}\langle c, t\rangle$. Finally this implies that $j = i$. ■

Linear algebra

Let K be a field, V a nontrivial finite dimensional K vector space, $\varphi \in \operatorname{End}_K(V)$, and $0 \neq v \in V$.

Recall, the **minimal polynomial** m_φ is the unique *monic*[3] generator of the principal ideal $I_\varphi := \{f \in K[x] \mid f(\varphi) = 0 \in \operatorname{End}_K(V)\}$, the so-called **vanishing ideal** of φ.

Analogously, the **minimal polynomial** $m_{\varphi,v}$ **with respect to** v is the unique *monic* generator of the principal ideal $I_{\varphi,v} := \{f \in K[x] \mid f(\varphi)(v) = 0 \in V\}$, the so-called **vanishing ideal of** φ **with respect to** v.

[1] German: Begleitmatrix

[2] German: Bahn

[3] German: normiert

Exercise 5.8. For $0 \neq v \in V$ let $U_{\varphi,v} := \langle \varphi^i(v) \mid i \in \mathbb{N}_0 \rangle \leq V$. Then

1. $m_{\varphi,v} = m_{\varphi_{|U_{\varphi,v}}}$.
2. $\dim_K U_{\varphi,v} = \min\{d \in \mathbb{N} \mid (v, \varphi(v), \ldots, \varphi^d(v)) \text{ are } K\text{–linearly dependent}\} \geq 1$.
3. $\deg m_{\varphi,v} = \dim_K U_{\varphi,v}$.
4. $m_\varphi = \operatorname{lcm}\{m_{\varphi,v} \mid 0 \neq v \in V\}$. This gives an algorithm to compute the minimal polynomial of φ as the lcm of at most ℓ minimal polynomials $m_{\varphi,v_1}, \ldots, m_{\varphi,v_\ell}$, where $\ell = \dim_K V$.
5. $\alpha \in \operatorname{End}_K(V)$ is an automorphism if and only if $m_\alpha(0) \neq 0 \in K$. This gives an algorithm to compute the inverse of α.

□

Definition 5.9. Let $0 \neq f \in K[x]$ and let $k \in \mathbb{N}$. We define the **order of** f: If $f(0) \neq 0$ we define

$$\operatorname{ord} f := \min\{k > 0 : f \mid x^k - 1\} \text{ or } \infty.$$

If $f(0) = 0$, then we write $f = x^r \bar{f}$ with $r \in \mathbb{N}$ minimal such that $\bar{f}(0) \neq 0$ and we define

$$\operatorname{ord} f := \operatorname{ord} \bar{f}.$$

□

Definition 5.10. Let $\alpha \in \operatorname{Aut}_K(V)$ and $\langle \alpha \rangle$ the cyclic subgroup of $\operatorname{Aut}_K(V)$ generated by α. By

$$\operatorname{ord} \alpha := |\langle \alpha \rangle|$$

denote the **order** of the group element α. For $v \in V$ denote the **orbit** of v under the action of the cyclic subgroup $\langle \alpha \rangle$ as usual by

$$\langle \alpha \rangle v := \{\alpha^i(v) \mid i \in \mathbb{Z}\}.$$

□

Proposition 5.11. Let $\alpha \in \operatorname{Aut}_K(V)$.

1. $\operatorname{ord} m_\alpha = \operatorname{ord} \alpha$.
2. $\operatorname{ord} m_{\alpha,v} = |\langle \alpha \rangle v|$ for $v \neq 0$.
3. If m_α is irreducible then $|\langle \alpha \rangle v| = \operatorname{ord} m_\alpha$ for all $v \neq 0$.
4. If K is finite and m_α irreducible then $\operatorname{ord} m_\alpha \mid (|V| - 1)$.
5. If there exists a vector $v \in V$ with $\langle \alpha \rangle v = V \setminus \{0\}$ then m_α is irreducible.

□

Proof.

1. Follows from the equivalence

$$\alpha^k = \mathrm{id}_V \iff (x^k - 1)(\alpha) = 0 \iff m_\alpha \mid x^k - 1.$$

2. If $|\langle\alpha\rangle v| < \infty$ then there exists $0 \leq i < j$ with $\alpha^i(v) = \alpha^j(v)$. Hence, $\alpha^{j-i}(v) = v$. Therefore (even if $|\langle\alpha\rangle v| = \infty$)
$$\begin{array}{rcl} |\langle\alpha\rangle v| & = & \min\{k > 0 \mid \alpha^k(v) = v\} \\ & = & \min\{k > 0 \mid (\alpha^k - \mathrm{id}_V)(v) = 0 \in V\} \\ & = & \min\{k > 0 : m_{\alpha,v} | x^k - 1\} \\ & = & \operatorname{ord} m_{\alpha,v}. \end{array}$$

3. If m_α is irreducible then the statements $m_{\alpha,v} \mid m_\alpha$ and $\deg m_{\alpha,v} > 0$ are equivalent to $m_{\alpha,v} = m_\alpha$. (2) completes the argument.

4. Follows from (3) which implies that the orbits of $\langle\alpha\rangle$ in $V \setminus \{0\}$ are all of the same length: For all $v, w \in V \setminus \{0\}$ $|\langle\alpha\rangle v| = |\langle\alpha\rangle w|$. It follows that $|V| = |K|^\ell$ for some $\ell \in \mathbb{N}$. Moreover, $|V| = 1 + k \cdot \operatorname{ord} m_\alpha$[4], i.e. $\operatorname{ord} m_\alpha \mid |V| - 1$.

5. First note that if $\langle\alpha\rangle v = V \setminus \{0\}$ for one $v \in V$ then also for all $v \in V \setminus \{0\}$ (being all elements of the orbit). In particular, assume that $U_{\alpha,v} \subsetneq V$. Then there exists $w \in V \setminus U_{\alpha,v}$, i.e. there exists $i < 0$ such that $\alpha^i(v) = w$ due to our initial note. Thus we can write $v = \alpha^{-i}(w)$ and get $U_{\alpha,v} \subsetneq U_{\alpha,w}$. All in all, we can assume that there exists $v \in V$ such that $U_{\alpha,v} = V$ and hence $m_{\alpha,v} = m_\alpha$ for all $v \neq 0$. If $m_\alpha = gh$ we want to prove that either $m_\alpha \mid g$ or $m_\alpha \mid h$: For $v \neq 0$ we obtain $0 = m_{\alpha,v}(\alpha)(v) = (gh)(\alpha)(v) = (g(\alpha) \circ h(\alpha))(v) = g(\alpha)(h(\alpha)(v))$. Hence, either $h(\alpha)(v) = 0$ or $v' := h(\alpha)(v) \neq 0$ and $g(\alpha)(v') = 0$. In other words, either $m_\alpha = m_{\alpha,v} \mid h$ or $m_\alpha = m_{\alpha,v'} \mid g$. ■

Period length

Convention 5.12. Let $V = K^{1\times\ell}$ and $t \in V \setminus \{0\}$. Identify $\mathrm{Aut}_K(V)$ with $\mathrm{GL}_\ell(K)$ in the obvious way. Viewing the regular matrix C (of Lemma 5.7.(2)) as an automorphism we get $\chi = m_C$. Define $\chi_t := m_{C,t}$. □

Corollary 5.13. With c and ℓ defined as above the following statements hold:

1. $\operatorname{per} c = \operatorname{ord} \chi$.
2. $\operatorname{per}\langle c, t\rangle = \operatorname{ord} \chi_t$ for $t \neq 0$.
3. If χ is irreducible then $\operatorname{per}\langle c, t\rangle = \operatorname{per} c$ for all $t \neq 0$.

And in case $K = \mathbb{F}_q$ (i.e., finite with q elements):

4. If χ is irreducible then $\operatorname{per} c \mid q^\ell - 1$.
5. If $\operatorname{per} c = q^\ell - 1$ then χ is irreducible.

□

Proof. All statements follow easily from Lemma 5.7 and Proposition 5.11. ■

Moreover, we can follow that there always exists a t such that the period length of c is reached.

[4] The $+1$ comes from $0 \in V$.

Corollary 5.14. Again, let c, t and ℓ defined as above. Then there exists a (row) vector $t \in K^{1\times\ell}$ with $\operatorname{per}\langle c, t\rangle = \operatorname{per} c$. □

Proof. We denote by C the companion matrix. Let $e_i = (0,\dots,0,1,0,\dots,0)$ denote the ith basis vector of $K^{1\times\ell}$. We compute the orbit of e_ℓ w.r.t. C:

$$\begin{array}{rcl} e_\ell \cdot C^0 & = & e_\ell \\ e_\ell \cdot C^1 & = & e_{\ell-1} + c_{\ell-1} e_\ell \\ (e_\ell \cdot C)\cdot C = e_\ell \cdot C^2 & = & e_{\ell-2} + c_{\ell-1} e_{\ell-1} + \left(c_{\ell-1}^2 + c_{\ell-2}\right) e_\ell \\ & \vdots & \end{array}$$

By the above construction it is clear that the orbit of e_ℓ w.r.t. C has length $|e_\ell\langle C\rangle| \geq \ell$. Now, due to the linear independency of the e_i and using notation from Exercise 5.8 we see that $U_{C,\ell} = V = K^{1\times\ell}$ due to dimension reasons. Thus by Exercise 5.8 (1) we get

$$m_{C,\ell} = m_{C_{|U_{C,\ell}}} = m_C.$$

We conclude with Corollary 5.13 (1) & (2):

$$\operatorname{per} c = \operatorname{ord}\chi = \operatorname{ord} m_C = \operatorname{ord} m_{C,\ell} = \operatorname{ord}\chi_\ell = \operatorname{per}\langle c, \ell\rangle.$$

■

Example 5.15. Let $K = \mathbb{F}_2$.

1. Consider the 3-bit LFSR (i.e., of degree $\ell = 3$) and maximum possible period length $q^\ell - 1 = 2^3 - 1 = 8 - 1 = 7$.

c^{tr}	χ	irred.	s	orbit lengths
$(1,0,0)$	x^3+1	false	100|100, 110|110, 1|111	$3+3+1$
$(1,1,0)$	x^3+x+1	true	1001011|100	7
$(1,0,1)$	x^3+x^2+1	true	1001110|100	7
$(1,1,1)$	x^3+x^2+x+1	false	1001|100, 01|010, 1|111	$4+2+1$

Note that, for example,

$$(x^2+x+1)(x+1) = x^3+x^2+x+x^2+x+1 = x^3+2x^2+2x+1 = x^3+1 \text{ over } \mathbb{F}_2.$$

2. Consider the 4-bit LFSR (i.e., of degree $\ell = 4$) and maximum possible period length $q^\ell - 1 = 2^4 - 1 = 16 - 1 = 15$.

c^{tr}	χ	irred.	s	orbit lengths
$(1,1,0,0)$	x^4+x+1	true	100010011010111|1000	15
$(1,0,0,1)$	x^4+x^3+1	true	100011110101100|1000	15
$(1,1,1,1)$	$x^4+x^3+x^2+x+1$	true	10001|1000, 01001|0100	$5+5+5$
			10100|1010	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

□

Definition 5.16. We call a linear recurrence equation **irreducible** if χ is irreducible. If moreover $K = \mathbb{F}_q$ is a finite field then we call the LRE **transitive** if $\operatorname{per} c = q^\ell - 1$, where ℓ is its degree. □

Remark 5.17. There are faster ways to compute $\operatorname{per} c$ and to decide the transitivity of LREs. Consider for example $c = (1,1,0,0)^{\mathrm{tr}}$ with $\chi = x^4 + x + 1$ in the above table. Since χ is irreducible we know from Corollary 5.13 that $\operatorname{ord} \chi \mid 15$. It is obvious that $\chi \nmid x^k - 1$ for $k = 1, 3, 5$ (these are the divisors of 15). Hence $\operatorname{per} c = \operatorname{ord} \chi = 15$, the maximal possible period length, i.e., the corresponding LFSR is transitive. □

Exercise 5.18. Classify all irreducible 4-bit LFSR. How many of them are transitive? □

Remark 5.19. The **Mersenne twister** [Wik17f] is a modern pseudorandom bit generator with an impressive period length of $\operatorname{per} c = 2^{19937} - 1 \approx 4.3 \cdot 10^{6001}$. Its name comes from the fact that $2^{19937} - 1$ is a Mersenne prime number. □

5.3 Finite fields

For a better understanding on how to get good pseudorandom bit generators let us recall some ideas from finite field theory.

Convention 5.20. In the following all fields are finite fields if not otherwise stated. □

Field extensions

Recall, if $K \leq L$ is a field extension, then L is a K-vector space. The degree of the field extension $K \leq L$ is defined as the dimension of L as a K-vector space:

$$[L : K] := \dim_K L.$$

For a 2-step field extension $K \leq L \leq M$ the **degree formula**

$$[M : K] = [M : L] \cdot [L : K]$$

is a direct consequence of the definition.

In what follows we only deal with **finite** field extensions $K \leq L$, i.e., where

$$d := [L : K] < \infty.$$

For any element $\alpha \in L$ the $d+1$ elements $1, \alpha, \ldots, \alpha^d$ are always K–**linearly dependent**, which leads us to the next definition:

Definition 5.21. Let $K \leq L$ be a finite field extension and let $\alpha \in L$. The unique *monic* generator of the **vanishing (principal) ideal** $I_{\alpha,K} := \{f \in K[x] \mid f(\alpha) = 0\}$ is called the **minimal polynomial of** α **over the ground field** K, and denoted by $m_{\alpha,K}$, or simply m_α, if no confusion can occur about the ground field K. □

Remark 5.22. In the above definition the field L can be replaced by a (finite dimensional) K-algebra L. This gives a common generalization of the two definitions m_φ and $m_{\alpha,K}$ above, where in the former case $L = \operatorname{End}_K(V)$. □

Remark 5.23. Let $K \leq L$ be a finite field extension of degree d and $\alpha \in L$. The minimal polynomial $m_\alpha = m_{\alpha,K}$ satisfies the following properties:

1. For $f \in K[x]$ it holds: $f(\alpha) = 0 \iff m_\alpha \mid f$.
2. m_α is irreducible in $K[x]$ and $1 \leq \deg m_\alpha \leq d$.
3. If an irreducible monic polynomial $f \in K[x]$ satisfies $f(\alpha) = 0$ then $f = m_\alpha$.

□

We now recall **Kronecker's construction** of field extensions:

Exercise 5.24. Let K be a field and $f \in K[x]$. The residue class K-algebra $L := K[x]/\langle f \rangle$ is a field if and only if f is irreducible. In this case $[L : K] = \deg f$ and $m_{\bar{x}} = m_{\bar{x},K} = f$, where $\bar{x} := x + \langle f \rangle \in K[x]/\langle f \rangle$. □

Example 5.25.

1. Let $f := x - a$ for $a \in K$. Then $K[x]/\langle f \rangle \cong K$.
2. Let K be a subfield of $\mathbb{R}$, e.g., $K = \mathbb{Q}$ or $K = \mathbb{R}$. Then $f = x^2 + 1$ is irreducible and the field $K[x]/\langle f \rangle$ is an extension of degree 2 over K with $(1, \bar{x})$ as a K-basis satisfying $\bar{x}^2 = -1$.
3. Let $K = \mathbb{F}_2$ and $f = x^3 + x + 1$. The degree 3 polynomial f is irreducible since it has no roots in its field of definition $\mathbb{F}_2$. The field $L := \mathbb{F}_2[x]/\langle f \rangle$ is an extension of degree 3 over $\mathbb{F}_2$ with $(1, \bar{x}, \bar{x}^2)$ as an $\mathbb{F}_2$-basis and elements

$$L = \{0, 1, \bar{x}, \bar{x}^2, 1 + \bar{x}, \bar{x} + \bar{x}^2, 1 + \bar{x} + \bar{x}^2, 1 + \bar{x}^2\}.$$

□

Order of field elements

Lemma 5.26. Let K be a field.

1. Let $f \in K[x]$ be irreducible and $f \neq x$. Then $\bar{x} \neq 0$ in $L := K[x]/\langle f \rangle$ and $\operatorname{ord} f = \operatorname{ord} \bar{x}$ in the multiplicative group $L^* := (L \setminus \{0\}, \cdot)$.
2. Let $K \leq L$ be a (finite) field extension and $\alpha \in L^*$. Then $\operatorname{ord} m_{\alpha,K} = \operatorname{ord} \alpha$ in the multiplicative group L^*.

□

Proof.

1. $\bar{x}^k = 1 \iff f \mid x^k - 1$.
2. $\alpha^k = 1 \iff \alpha$ is a root of $x^k - 1 \iff m_{\alpha,K} \mid x^k - 1$. ■

Corollary 5.27. Let $K = \mathbb{F}_q$ be a finite field with q elements and $f \in \mathbb{F}_q[x]$ of degree n. Then

1. f irreducible $\implies \operatorname{ord} f \mid q^n - 1$.

2. $\operatorname{ord} f = q^n - 1 \implies f$ irreducible.

□

Proof. We can assume that f is monic.

First assume that $f(0) \neq 0$: Take $V = K^{1\times n}$ and $\beta \in \operatorname{Aut}_K(V)$ with $m_\beta = f$ (e.g., $\beta : t \mapsto t \cdot C$, where C is the companion matrix of f, which is due to $f(0) \neq 0$ regular). Now apply Proposition 5.11 (4) and (5) for both statements.

Now we assume that $f(0) = 0$:

1. $f = x$ is the only irreducible monic polynomial with $f(0) = 0$. By definition $\operatorname{ord} x \overset{5.9}{:=} \operatorname{ord} 1 = 1$.

2. Now let $f = x^r \bar{f}$ for some $r \geq 1$, i.e. f is not irreducible. By construction $\deg \bar{f} = n - r$ and $\bar{f}(0) \neq 0$. By Definition 5.9 we know that $\operatorname{ord} f = \operatorname{ord} \bar{f}$. If $\bar{f}$ is irreducible we know by (1) that $\operatorname{ord} \bar{f} \mid q^{n-r} - 1$. Otherwise we consider the product $\bar{f} = \prod f_i$ and do the same argument for the irreducible factors f_i. Glueing this information back together (a bit technical) we get an estimate for $\operatorname{ord} \bar{f} < q^n - 1$. All in all, it follows that $\operatorname{ord} f < q^n - 1$. ■

Definition 5.28. Let $K = \mathbb{F}_q$ be a finite field with q elements and L a finite field extension of $\mathbb{F}_q$.

1. A degree n polynomial $f \in \mathbb{F}_q[x]$ is called **primitive** if $\operatorname{ord} f = q^n - 1$. Primitive polynomials are *irreducible* by the above corollary.

2. An element $\alpha \in L^*$ is called **primitive**[5] if $\operatorname{ord} \alpha = |L^*|$, i.e., if $L^* = \langle \alpha \rangle := \{\alpha^i \mid i \in \mathbb{N}_0\}$.

□

Lemma 5.29. Let $K = \mathbb{F}_q$ a finite field with q elements.

1. Let $f \in \mathbb{F}_q[x]$ be a primitive polynomial and $L := \mathbb{F}_q[x]/\langle f \rangle$. Then $\bar{x}$ is a primitive element of L.

2. If $\mathbb{F}_q \leq L$ is a finite field extension then $\alpha \in L^*$ is primitive iff $m_{\alpha,K}$ is a primitive polynomial (of degree $[L : K]$).

□

Proof.

1. Define $n := \deg f$. Then $|L| = q^n$ and $|L^*| = q^n - 1$. Now use that $\operatorname{ord} \bar{x} \overset{5.26.(1)}{=} \operatorname{ord} f = q^n - 1$.

2. Define $n := [L : K]$. If $L^* = \langle \alpha \rangle$ then $\operatorname{ord} \alpha = |L^*| = q^n - 1$. Set $f = m_{\alpha,K}$. Lemma 5.26.(2) implies that $\operatorname{ord} f = \operatorname{ord} \alpha = q^n - 1$. Using that $\deg f \leq n$ and that $\operatorname{ord} f \mid q^{\deg f} - 1$ (Corollary 5.27.(1)) we conclude that $n = \deg f$ and finally the primitivity of $f = m_{\alpha,K}$. ■

Exercise 5.30. Let $L := \mathbb{F}_2[x]/\langle f \rangle$ and

1. $f := x^3 + x + 1$. Prove that f is a primitive polynomial, or equivalently, that $\bar{x} \in L$ is a primitive element, i.e., $L^* = \langle \bar{x} \rangle$.

[5] Unfortunately, this name conflicts the notion of primitive elements of (algebraic) field extensions.

2. $f := x^4 + x^3 + x^2 + x + 1$. First prove that L is a field. Prove that f is an imprimitive polynomial, or equivalently, that $\bar{x} \in L$ is an imprimitive[6] element, i.e., $L^* \neq \langle \bar{x} \rangle$

□

Some field theory

Let K be a field. Recall:

1. $K[x]$ is a **Gaussian domain**[7]. A more suggestive name is **unique factorization domain (UFD)** or simply **factorial domain**[8].
2. For $f \in K[x]$ the following holds:
   a) $f(a) = 0 \iff (x - a) \mid f$.
   b) $f(a) = f'(a) = 0 \iff (x - a)^2 \mid f$, where f' is the derivative of f w.r.t. x.
3. The **characteristic** of K is defined as $\operatorname{char} K = \min\{c \in \mathbb{N} \mid c \cdot 1 = 0\}$ or 0. i.e., it is the unique nonnegative generator of the principal ideal $\ker(\mathbb{Z} \to K, c \mapsto c \cdot 1) \lhd \mathbb{Z}$. $\operatorname{char} K$ is either zero or a prime number.
4. If $\operatorname{char} K = p > 0$ then $\mathbb{F}_p \leq K$. Else $\mathbb{Q} \leq K$. The fields $\mathbb{F}_p \cong \mathbb{Z}/p\mathbb{Z}$ resp. $\mathbb{Q} = \operatorname{Quot}(\mathbb{Z})$ are therefore called **prime fields**. Each field contains exactly one prime field as the smallest subfield.
5. For a finite field extension $K \leq L$ define for an element $\alpha \in L$ the smallest subring of L containing K and α
$$K[\alpha] := \left\{ \sum_{i=1}^{n} \lambda_i \alpha^i \mid n \in \mathbb{N}_0, \lambda_i \in K \right\}.$$
6. The vanishing ideal $I_{\alpha,K} = \langle m_{\alpha,K} \rangle$ is the kernel of the ring epimorphism $K[x] \to K[\alpha]$, $x \mapsto \alpha$. Hence $K[\alpha] \cong K[x]/\langle m_{\alpha,K} \rangle$ as K-algebras and $K(\alpha) := K[\alpha]$ is a *field* with $[K(\alpha) : K] = \deg m_{\alpha,K}$. The field $K(\alpha)$ is called the **intermediate field**[9] generated by α.

Example 5.31. Let $L := \mathbb{F}_2[x]/\langle x^4 + x^3 + x^2 + x + 1 \rangle$ as in Exercise 5.30.(2). The element $\alpha := \bar{x}^3 + \bar{x}^2 + 1$ satisfies $\alpha^2 = \bar{x}^3 + \bar{x}^2$. Hence $m_{\mathbb{F}_2,\alpha} = x^2 + x + 1$ and $\mathbb{F}_2(\alpha) = \mathbb{F}_2[\alpha]$ is an intermediate field of degree $[\mathbb{F}_2(\alpha) : \mathbb{F}_2] = 2$: $K(\alpha) = K + K\alpha = \{0, 1, \alpha, 1 + \alpha\}$. □

Proposition 5.32. Let $K \leq L$ be a field extension and let $f \in K[x]$ be a monic irreducible polynomial with $f(\alpha) = f(\alpha') = 0$ for two elements $\alpha, \alpha' \in L$. Then $K(\alpha) \cong K(\alpha')$ as K-algebras (or as fields *over* K). □

Proof. $f(\alpha) = 0$ and f monic irreducible implies $f = m_{\alpha,K}$ by Remark 5.23.(3). The same ist true for α'. Hence $K(\alpha) \cong K[x]/\langle f \rangle \cong K(\alpha')$. ■

Now we recall the notion of the splitting field of a polynomial $f \in K[x]$.

[6] Although $\bar{x}$ is a primitive element of the field extension $\mathbb{F}_2 \leq \mathbb{F}_2[\bar{x}] = \mathbb{F}_2[x]/\langle x^4 + x^3 + x^2 + x + 1 \rangle$.

[7] German: Gaußcher Bereich

[8] German: Faktorieller Bereich

[9] German: Zwischenkörper

Definition 5.33. Let $K \leq L$ and $f \in K[x]$. We define the following properties:

1. f **splits over**[10] L if f splits as a product of linear factors (when viewed as a polynomial) over $L[x]$.
2. L is a **splitting field**[11] of f over K if f splits over L and L is minimal with this property.

□

Remark 5.34. If f splits over L with roots $\alpha_1, \ldots, \alpha_n$ then $K(\alpha_1, \ldots, \alpha_n) = K[\alpha_1, \ldots, \alpha_n]$ is a splitting field of f contained in L. □

Theorem 5.35. For each $f \in K[x]$ there exists a splitting field, unique up to K-isomorphism. □

Proof. The above remark shows that it is enough to construct a field $M \geq K$ over which f splits. We may assume[12] that f is irreducible (otherwise we do the following for all factors). The field $L := K[x]/\langle f \rangle$ contains (at least) one root of f which is $\alpha := \bar{x}$. Hence $f = (x - \alpha)\bar{f} \in L[x]$. Proceed by induction on $\deg f$ resp. $\deg \bar{f}$. ■

It follows that we can talk about *the* splitting field of f over K. Recall, $\operatorname{char} K = p > 0$ means that $p\alpha = 0$ for any α in any field extension $L \geq K$.

Lemma 5.36. Let K be a field with $\operatorname{char} K = p > 0$. For each $i \in \mathbb{N}_0$ the map $\varphi_i : K \to K, x \mapsto x^{p^i}$ is an automorphism of K which fixes the prime field $\mathbb{F}_p$. It is called the i**-th Frobenius automorphism** of K. □

Proof. Since $\varphi_i = \varphi_1^i$ it suffices to consider $i = 1$. Of course $1^p = 1$ and $(\alpha\beta)^p = \alpha^p\beta^p$ in any field of any characteristic. We therefore only need to prove the "characteristic p formula"

$$(\alpha \pm \beta)^p = \alpha^p \pm \beta^p.$$

Indeed the **binomial theorem** $(\alpha+\beta)^p = \sum_{k=0}^{p} \binom{p}{k} \alpha^k \beta^{p-k}$ implies the statement since $\binom{p}{0} = \binom{p}{p} = 1$ and $p \mid \binom{p}{k}$ for $0 < k < p$. Proving the bijectiveness is an exercise. ■

Theorem 5.37. Let K be a field of prime characteristic p, $n \in \mathbb{N}$, and $q := p^n$. Consider $f := x^q - x \in \mathbb{F}_p[x]$.

1. f splits over K if and only if K contains exactly one subfield with q elements.
2. K is the splitting field of f if and only if $|K| = q$.

□

Proof. Set

$$N := \{\alpha \in K \mid f(\alpha) = 0\}.$$

[10] German: zerfällt über

[11] German: Zerfällungskörper

[12] It is not clear how to make this step constructive. From the constructive point of view, we have just assumed that we have an algorithm to factor polynomials in irreducible factors.

1. By construction $|N| \leq q$. Since f has no multiple roots ($f' = -1 \implies \gcd(f, f') = 1$) we conclude that f splits over $K \iff |N| = q$. The previous lemma implies that N is a subfield of K: $\alpha, \beta \in N \implies (\alpha - \beta)^q = \alpha^q - \beta^q = \alpha - \beta$ and we are done with the forward implication in (1).
   Now let $M \leq K$ be a subfield with q elements. Since $|M^*| = q - 1$ it follows that every $\alpha \in M^*$ is a root of $f = x(x^{q-1} - 1)$, hence $M \leq N$. From $|N| \leq q$ we conclude that $M = N$. This proves the uniqueness of $M = N$ and that f splits over a field K containing N. This proves statement (1).
2. (2) follows from (1) and the minimality of the splitting field. ■

Corollary 5.38.

1. If K is a finite field then $\operatorname{char} K = p > 0$ and $|K| = p^n$ for a prime p.
2. For each prime power $q = p^n$ there exists up to $\mathbb{F}_p$-isomorphism exactly one field with q elements.

□

Proof.

1. Since K is finite its characteristic $\operatorname{char} K = p > 0$ is prime. Hence, $\mathbb{F}_p$ is the prime field of K and, in particular, K is an $\mathbb{F}_p$-vector space. So $|K| = p^n$, where $n = [K : \mathbb{F}_p] := \dim_{\mathbb{F}_p} K$.
2. Follows from Theorem 5.37.(2) and the uniqueness of the splitting field applied to the polynomial $f = x^q - x$. ■

We have been referring to this field as $\mathbb{F}_q$. Now we can say "the field $\mathbb{F}_q$".

Corollary 5.39. The finite field $\mathbb{F}_q = \mathbb{F}_{p^n}$ contains the unique subfield (isomorphic to) $\mathbb{F}_{p^d}$ if and only if $d \mid n$. I.e.

$$\mathbb{F}_{p^d} \leq \mathbb{F}_{p^n} \iff d \mid n.$$

□

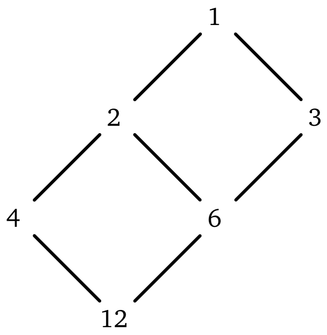
In other word, the **subfield lattice**[13] of $\mathbb{F}_{p^n}$ is isomorphic to the lattice of divisors of n (regardless of the prime number p).

Proof. Let $K \leq \mathbb{F}_q = \mathbb{F}_{p^n}$. Then K has characteristic p and the prime field $\mathbb{F}_p$ of $\mathbb{F}_q$ is the prime field of K. Hence $K = \mathbb{F}_{p^d}$ for some $1 \leq d \leq n$. The degree formula $n = [\mathbb{F}_{p^n} : \mathbb{F}_p] = [\mathbb{F}_{p^n} : \mathbb{F}_{p^d}]\underbrace{[\mathbb{F}_{p^d} : \mathbb{F}_p]}_{d}$ implies that $d \mid n$.
Now we prove the converse. Let $d \mid n$. First note that $\alpha^{p^d} = \alpha$ implies $\alpha^{p^n} = \alpha$. In particular, the roots of $x^{p^d} - x$ are also roots of $x^{p^n} - x$. So $x^{p^d} - x$ splits over $\mathbb{F}_{p^n}$. Theorem 5.37.(1) then states that $\mathbb{F}_{p^n}$ contains the unique field with p^d elements. ■

Example 5.40.

[13] German: Zwischenkörperverband

1. $\mathbb{F}_4 \not\leq \mathbb{F}_8$, but $\mathbb{F}_4 < \mathbb{F}_{16}$.
2. The subfield lattice of $\mathbb{F}_{p^{12}}$ is isomorphic to the divisor lattice of 12



□

Irreducible polynomials over finite fields

With our previous investigations we know now that we can construct the finite field $\mathbb{F}_{p^n}$ as the splitting field of $x^{p^n} - x$. This would eventually involve iterated Kronecker constructions. So it is natural to ask if we can get the job done with just one Kronecker construction. This question is equivalent to asking if there exists an irreducible polynomial of degree n over $\mathbb{F}_p$.

Lemma 5.41. Let K be a field and $f \in K[x]$ with $f(0) \neq 0$. Then $\operatorname{ord} f \mid k \iff f \mid x^k - 1$. □

Proof. Exercise. ■

Corollary 5.42. Let $K = \mathbb{F}_q$ with $q = p^m$.

1. Each irreducible polynomial $f \in \mathbb{F}_q[x]$ with $\deg f = n$ is **square free**[14,15] and splits over $\mathbb{F}_{q^n}$.
2. $\mathbb{F}_q[x]/\langle f \rangle \cong \mathbb{F}_{q^n}$ for all irreducible $f \in \mathbb{F}_q[x]$ with $\deg f = n$.

□

Proof.

1. Corollary 5.27 (1) states that $\operatorname{ord} f \mid q^n - 1$ which is equivalent to $f \mid x^{q^n - 1} - 1$ by Lemma 5.41. Multiplying by x, it also holds that $f \mid x^{q^n} - x = x(x^{q^n-1} - 1)$. But the polynomial $x^{q^n} - x$ splits with distinct roots over $\mathbb{F}_{q^n}$ by Theorem 5.37 and its proof (applied to q^n). The same holds for the divisor f.
2. The statement follows from $|\mathbb{F}_q[x]/\langle f \rangle| = q^n$, Theorem 5.37 (2) and Corollary 5.38. ■

We introduce a shorthand notation for the number of irreducible monic polynomials.

[14] German: quadratfrei

[15] i.e., it has no multiple roots over its splitting field.

Notation 5.43. Set

$$A(d) = A(d,q) := |\{f \in \mathbb{F}_q[x] : f \text{ irreducible monic with } \deg f = d\}|.$$

□

For the following theorem we need a small preparation:

Lemma 5.44. Let π be the the product of all monic irreducible polynomials f over $\mathbb{F}_q$ with $\deg f \mid n$. Then $\pi = x^{q^n} - x$. □

Proof. By Corollaries 5.39 and 5.42 we know that each monic irreducible polynomial f with $\deg f \mid n$ fulfills $f \mid x^{q^n} - x$. Since we are working over a principal ideal domain and, thus, all these f are also prime we know that $\pi \mid x^{q^n} - x$. Now, using again Corollary 5.39 we get $\pi = x^{q^n} - x$. ■

Theorem 5.45. For $K = \mathbb{F}_q$ the numbers $A(d)$ satisfy

$$\sum_{d|n} dA(d) = q^n. \tag{5.2}$$

In particular, $A(1) = q$ and $A(d) = \frac{q^d - q}{d}$ if d is prime. □

Proof. Set $L := \mathbb{F}_{q^n}$. We know that $\mathbb{F}_{q^d} \leq L \iff d \mid n$. First note that by Lemma 5.44 $x^{q^n} - x$ is the product of all monic irreducible polynomials $f \in \mathbb{F}_q[x]$ with $\deg f \mid n$. By counting the degrees we directly get the formula $\sum_{d|n} dA(d) = q^n$ as each such polynomial of degree d contributes d to the total degree.
If d is prime, note that there exists no intermediate subfield between $\mathbb{F}_{q^d}$ and $\mathbb{F}_q$ as the field extension has prime degree d. Thus every monic irreducible polynomial dividing $x^{q^d} - x$ has either degree d or degree 1. There exist exactly q linear such polynomials. Again summing up all such polynomials we have $A(d)$ monic irreducible polynomials of degree d, we can again count the degree:

$$dA(d) + q = q^d.$$

This finishes the proof. ■

Example 5.46. Let $q = 2$. Now we list all minimal polynomials (= irreducible monic polynomials) together with their degrees for the following fields:

1. $K = \mathbb{F}_{2^2}$

$m_{\alpha,\mathbb{F}_2}$	x	$x+1$	x^2+x+1	
deg	1	1	2	$\sum = 4$

2. $K = \mathbb{F}_{2^4}$

$m_{\alpha,\mathbb{F}_2}$	x	$x+1$	x^2+x+1	x^4+x+1	x^4+x^3+1	$x^4+x^3+x^2+x+1$	
deg	1	1	2	4	4	4	$\sum = 16$

□

Remark 5.47. We list the following facts without proof:

1. Asymptotically: $A(d) \sim \frac{q^d}{d}$.
2. Since Equation 5.2 is an inclusion-exclusion counting formula over a lattice one can use the **Möbius function**
$$\mu(d) := \begin{cases} 1 & , \quad d = 1 \\ 0 & , \quad d \text{ is not square free} \\ (-1)^k & , \quad d \text{ is the product of } k \text{ distinct primes} \end{cases}$$
to "invert" it:
$$A(n) = A(n,q) = \frac{1}{n} \sum_{d|n} \mu(d) q^{\frac{n}{d}}.$$

□

Example 5.48. $A(20,q) = \frac{1}{20}(q^{20} - q^{10} - q^4 + q^2)$. □

Primitive polynomials

Counting primitive polynomials is much simpler.

Proposition 5.49. Let K be a field and U a *finite* subgroup of K^*. Then U is cyclic. □

Proof. Exercise. ■

Corollary 5.50. Let φ denote **Euler's totient function**:[16]. Let $q = p^d$.

1. There are exactly $\varphi(q-1)$ primitive elements in $\mathbb{F}_q^*$.
2. There are exactly $\frac{\varphi(q-1)}{d}$ primitive monic polynomials of degree d in $\mathbb{F}_p[x]$.

□

Proof.

1. Proposition 5.49 implies that the multiplicative group $\mathbb{F}_q^* = \{a^0, a^1, \ldots, a^{q-2}\}$ is cyclic. In particular, a^i is primitive $\overset{\text{Def 5.28}}{\iff} \mathbb{F}_q^* = \{(a^i)^k \mid k \in \mathbb{N}_0\} \iff \gcd(i, q-1) = 1$.
2. Every primitive $f \in \mathbb{F}_p[x]$ of degree d is the minimal polynomial of exactly d primitive elements in $L = \mathbb{F}_{p^d} = \mathbb{F}_q$. This follows from the irreducibility of f (Corollary 5.27.(2)) and Lemma 5.29 using (1) and the same argument as in the proof of Theorem 5.45. ■

Example 5.51. In the following two examples we mainly sum up computations we did before:

[16] German: Recall: $\varphi(n) = |(\mathbb{Z}/n\mathbb{Z})^*| = |\{i | 0 \leq i \leq n \text{ s.t. } \gcd(i,n) = 1\}| = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$ for distinct primes p dividing n. φ is multiplicative, $\varphi(mn) = \varphi(m)\varphi(n)$ if $\gcd(m,n) = 1$. For p prime and $k \geq 1$ we have $\varphi(p^k) = p^k - p^{k-1} = p^k\left(1 - \frac{1}{p}\right)$.

1. For $p = 2$ and $d = 2$ we get $q = 2^2 = 4$. $\mathbb{F}_4 = \mathbb{F}_{2^2} = \{0, 1, \omega, 1 + \omega\}$ with $\omega^2 + \omega + 1 = 0$. ω and $1 + \omega$ are all primitive elements of $\mathbb{F}_4$ and their minimal polynomial $x^2 + x + 1$ the only irreducible and primitive polynomial of degree 2 over $\mathbb{F}_2$.

2. Let $q = 16$, so we could have $p = 2$ and $d = 4$ or $p = 4$ and $d = 2$. There are $\varphi(16 - 1) = 8$ primitive elements in $\mathbb{F}_{16} = \mathbb{F}_{2^4} = \mathbb{F}_{4^2}$. Hence there are $\frac{8}{2} = 4$ primitive polynomials

   $$x^2 + x + \omega, \quad x^2 + x + \omega^2, \quad x^2 + \omega x + \omega, \quad x^2 + \omega^2 x + \omega^2$$

   of degree 2 over $\mathbb{F}_4$ and $\frac{8}{4} = 2$ primitive polynomials

   $$x^4 + x + 1, \quad x^4 + x^3 + 1$$

   of degree 4 over $\mathbb{F}_2$. The polynomial $x^4 + x^3 + x^2 + x + 1 = \frac{x^5 - 1}{x - 1}$ is the only irreducible imprimitive polynomial of degree 4 over $\mathbb{F}_2$.

□

Example 5.52. We compare $A(d)$ and the number of primitive polynomials of degree d over $\mathbb{F}_2$:

d	1	$\underline{2}$	$\underline{3}$	4	$\underline{5}$	6	$\underline{7}$	8	9	10	$\underline{11}$	16
$A(d, 2)$	2	1	2	3	6	9	18	30	56	99	186	4080
primitive	2	1	2	2	6	6	18	16	48	60	176	2048

□

We will be using primitive polynomials χ over finite fields to construct pseudorandom sequences $s = \langle c, t\rangle$ of maximal possible period lengths (cf. Definition 5.6). Since $\chi = \chi_c$ will be part of the secret key, we need to know how to randomly choose primitive polynomials. The idea will be to randomly choose a polynomial and then to test its primitiveness.

5.4 Statistical tests

Let X_n and U_n denote random variables with values in $\{0, 1\}^n$ where U_n is the *uniformly distributed* one. Note that any map $f : \{0, 1\}^n \to \{0, 1\}^m$ induces a random variable $Y_m := f \circ X_n$.

Example 5.53. Define linear pseudorandom bit **generator** G as the map

$$G : \left\{ \begin{array}{ccc} \{0,1\}^{2\ell} & \to & \{0,1\}^{\bullet} \\ (c, t) & \mapsto & \langle c, t\rangle \end{array} \right. ,$$

where we consider the pair (c, t) as an ℓ-bit LFSR given by $c \in \mathbb{F}_2^{\ell \times 1}$ and initial value $t \in \mathbb{F}_2^{1 \times \ell}$. By truncation to the first n bits we get a map

$$G_n : \left\{ \begin{array}{ccc} \{0,1\}^{2\ell} & \to & \{0,1\}^{n} \\ (c, t) & \mapsto & \langle c, t\rangle_{i=0,\dots,n-1} \end{array} \right.$$

Define the random variable

$$X := G \circ U_{2\ell}$$

In other words, X is a (linear) pseudorandom bit **generator** with *random* seed. Define the finite random variable

$$X_n := G_n \circ U_{2\ell}$$

with values in $\{0,1\}^n$. □

Our goal will be to compare X_n with U_n.

Definition 5.54. A **(polynomial) statistical test** is a polynomial probabilistic algorithm

$$A : \left\{ \begin{array}{ccc} \{0,1\}^\bullet & \to & \{0,1\} \\ s & \mapsto & A(s) \end{array} \right. .$$

We say that $s \in \{0,1\}^\bullet$ **passes** the test A if $A(s) = 1$. □

Remark 5.55. The composition $A \circ X_n$ is a random variable with values in $\{0,1\}$ for any random variable X_n (with values in $\{0,1\}^n$). $\mu_{A\circ X_n}(1)$ is the probability that an $s \in \{0,1\}^n$ that was chosen according to X_n passes the test A. □

The idea is to construct statistical tests A where $s \in \{0,1\}^n$ only passes A if it was chosen randomly, i.e., according to U_n.

Statistical randomness

The idea is to choose a **statistic**

$$S : \{0,1\}^\bullet \to \mathbb{R}$$

such that the distribution of $S \circ U_n$ converges to a *continuous* probability density f. A **continuous**[17] real valued random variable $X : \Omega \to \mathbb{R}$ has a **probability density** $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ satisfying the probability $P_X(I) = P(X \in I) = \int_I f(x)dx$ for any interval $I \subset \mathbb{R}$.

Example 5.56. For $\alpha \in (0,1)$ choose an interval $I \subset \mathbb{R}$ (as small as possible) with $\int_I f(x)dx > 1-\alpha$, or, equivalently, $\int_{\mathbb{R}\setminus I} f(x)dx < \alpha$. Define the statistical test $A := A_{S,\alpha}$ **induced by the statistic** S by setting

$$A_{S,\alpha}(s) := \left\{ \begin{array}{lll} 1 & \text{if} & S(s) \in I \\ 0 & \text{if} & S(s) \notin I \end{array} \right. .$$

Then $\mu_{A\circ U_n}(1) > 1-\alpha$ and, equivalently, $\mu_{A\circ U_n}(0) < \alpha$. The real number α is called the **significance level**[18]. □

Recall that the expected value of the real valued random variable X with density f can be computed as $E(X) := \int_{x\in\mathbb{R}} xf(x)dx$. The **variance**[19] is defined by

$$\operatorname{Var}(X) := E((X - E(X))^2) = E(X^2) - E(X)^2.$$

Remark 5.57. Let X, Y be two (finite) random variables and $a, b, c \in \mathbb{R}$.

[17] German: stetig (hat zwei Bedeutungen!)

[18] German: Signifikanzniveau

[19] German: Varianz

1. E is linear, i.e.
$$E(aX + bY) = aE(X) + bE(Y).$$

2. If X and Y are *independent* then
$$\operatorname{Var}(aX + bY + c) = a^2 \operatorname{Var}(X) + b^2 \operatorname{Var}(Y).$$

3. Assume that Y is discrete and uniformly distributed, and that $(Y_1, \dots, Y_n)$ is the n-fold independent repetition of the experiment Y. Then $Z_n := \sum_{i=1}^n \left(\frac{Y_i - E(Y)}{\sqrt{n \operatorname{Var}(Y)}} \right) = \frac{\sum_{i=1}^n Y_i - nE(Y)}{\sqrt{n \operatorname{Var}(Y)}}$ converges to the **standard normal distribution**[20] $N(0,1)$ with expected value 0 and variance 1 (see Appendix A.1).

Proof. $E(Z_n) = 0$ due to the linearity of E.
$\operatorname{Var}(Z_n) = \frac{1}{n\operatorname{Var}(Y)} \operatorname{Var}\left(\sum_{i=1}^n Y_i - nE(Y)\right) = \frac{1}{n\operatorname{Var}(Y)} \operatorname{Var}\left(\sum_{i=1}^n Y_i\right) = \frac{n\operatorname{Var}(Y)}{n\operatorname{Var}(Y)} = 1.$ ■

□

Example 5.58. We now give two examples of polynomial statistical tests.

1. **Monobit (or balance) test**: Since $E(U_1) = \frac{1}{2}$ and $\operatorname{Var}(U_1) = \frac{1}{4}$ we define the statistic $S : \{0,1\}^\bullet \to \mathbb{R}$
$$S(s_0 s_1 \cdots s_{n-1}) := \frac{\sum_i s_i - \frac{n}{2}}{\sqrt{\frac{n}{4}}} = \frac{2\sum_i s_i - n}{\sqrt{n}}.$$
according to Remark 5.57.(3). Hence $S \circ U_n$ is an approximation of $N(0,1)$ for n large. For a given significance level $\alpha \in (0,1)$ choose $I = (-x, x)$ with $\operatorname{erfc}(\frac{x}{\sqrt{2}}) < \alpha$, i.e., $x < \sqrt{2}\operatorname{erfc}^{-1}(\alpha)$. Define
$$A : \begin{cases} \{0,1\}^n & \to \{0,1\} \\ s & \mapsto \begin{cases} 1 & \text{if } |S(s)| < x \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } |\sum_i s_i - \frac{n}{2}| < d \\ 0 & \text{otherwise} \end{cases} \end{cases},$$
where $d := \sqrt{\frac{n}{2}} \operatorname{erfc}^{-1}(\alpha)$. Then $\mu_{A \circ U_n}(1) > 1 - \alpha$ and, equivalently, $\mu_{A \circ U_n}(0) < \alpha$. For example, a bit sequence of length $n = 20000$ passes the monobit test with significance level $\alpha = 10^{-6}$ if the number of ones lies in the interval $(\frac{n}{2} - d, \frac{n}{2} + d) \approx (9654, 10346)$.

2. **Autocorrelation test**: The autocorrelation test on the bit sequence $s = (s_i) \in \mathbb{F}_2^{\mathbb{N}_0}$ with **distance** d is the monobit test on the bit sequence $s' = (s_i + s_{i+d}) \in \mathbb{F}_2^{\mathbb{N}_0}$.

3. There are many more such tests. See, for example, the so-called **runs test** [Wik16f].

□

Remark 5.59. LFSR have good statistical properties; in particular, their output passes all statistical tests in the above example. Indeed, if A is the monobit test then $\mu_{A \circ X_n}(1) \approx 1$ for $X_n = G_n \circ U_{2\ell}$ (cf. Example 5.53). □

Sketch of Proof. Each period of a primitive ℓ-bit LFSR (with maximal possible period length $2^\ell - 1$) consists of exactly $2^{\ell-1} - 1$ zeros and $2^{\ell-1}$ ones. ■

[20] German: Standard-Normalverteilung

Unpredictability

Passing all statistical randomness tests is not enough for a pseudorandom bit generator to be cryptographically secure. It must be "unpredictable" as well.

Remark 5.60 (Predictability of an LFSR). Let $s = \langle c, t\rangle$ be the output of an ℓ-bit LFSR of which 2ℓ consecutive bits are known, say $s_0, \dots, s_{2\ell-1}$, w.l.o.g. Hence the ℓ consecutive vectors $t^{(0)}, \dots, t^{(\ell-1)} \in \mathbb{F}_2^{1\times\ell}$ are known. They satisfy the equation

$$\begin{pmatrix} t^{(0)} \\ \vdots \\ t^{(\ell-1)} \end{pmatrix} \cdot c = \begin{pmatrix} s_\ell \\ \vdots \\ s_{2\ell-1} \end{pmatrix}.$$

This inhomogeneous linear system is solvable by our assumption on s. And there exists a unique solution for c if and only if $t^{(0)}, \dots, t^{(\ell-1)}$ are $\mathbb{F}_2$-linearly independent. In this case the next-bit $s_{2\ell} = (s_\ell \cdots s_{2\ell-1}) \cdot c$ can be predicted. This last condition is satisfied when the LFSR is irreducible and $t^{(0)} \neq 0$. □

Example 5.61. For $\ell = 4$ and $s = 10101111?? \dots$ we solve

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \cdot c = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

and obtain the unique solution $c = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$. The complete sequence is 101011110001001|1010 (cf. Example 5.15): We get $\chi_c = x^4 + x + 1$ with period length $\operatorname{ord} \chi_c = 2^4 - 1 = 15$. □

Definition 5.62. Let P be a statistical test. The **next-bit test with predictability** P is the statistical test $A := A_P$ defined by

$$s = s_0 \cdots s_n \mapsto A_P(s) = \begin{cases} 0 & \text{if} \quad P(s_0 \cdots s_{n-1}) = s_n \\ 1 & \text{if} \quad P(s_0 \cdots s_{n-1}) \neq s_n \end{cases}.$$

In other words, 0 for correct prediction and 1 for incorrect prediction. Note that if P is polynomial then so is A_P. □

Example 5.63 (Linear predictability of an LFSR). Define the statistical test P by setting

$$P(s) := (s_\ell \cdots s_{2\ell-1}) \cdot c$$

where $s = s_0 \cdots s_{2\ell-1} \in \{0,1\}^{2\ell}$ and c computed as in Remark 5.60 (a not necessarily unique solution). Remark 5.60 implies[21] that $\mu_{A_P \circ X_{2\ell+1}}(1) = 0$ for $X_{2\ell+1} = G_{2\ell+1} \circ U_{2\ell}$ (cf. Example 5.53). It follows that an LFSR is (linearly) predictable and in its original form cryptographically insecure. □

[21] We skipped some details here. For example, since P is defined only for $|s|$ even, A_P is a priori only defined for $|s|$ odd. To define it for $|s| = 2r$ we set $A_P(s) := A_P(s_0 \dots s_{2r-2})$. The non-uniqueness of c has to be addressed as well (cf. [Wik16a].)

5.5 Cryptographically secure pseudorandom bit generators

Again, let X_n and U_n be random variables with values in $\{0,1\}^n$ where U_n is the uniformly distributed one.

Definition 5.64.

1. A polynomial *deterministic* algorithm $G : \{0,1\}^\bullet \to \{0,1\}^\bullet$ is called a **pseudorandom bit generator (PRBG)** if there is a function $n : \mathbb{N} \to \mathbb{N}$ with $n(k) > k$ and $G(\{0,1\}^k) \subset \{0,1\}^{n(k)}$ for all $k \in \mathbb{N}$. The function n is called the **stretch function of** G.
2. A function $f : \mathbb{N} \to \mathbb{R}$ is called **negligible**[22] if for each positive polynomial p there exists a $k_0 \in \mathbb{N}$ such that $|f(k)| < \frac{1}{p(k)}$ for all $k \geq k_0$. The function $f(k) = e^{-k}$ is a prominent example of a negligible function.
3. We say that G **passes** the statistical test A if
$$k \mapsto \mu_{A \circ G \circ U_k}(1) - \mu_{A \circ U_{n(k)}}(1)$$
is negligible.
4. G is called **cryptographically secure (CSPRBG)** if G passes *all* polynomial statistical tests.

□

In other words, G is cryptographically secure if an adversary with limited computational resources has no non-negligible advantage in predicting the next bit of the pseudorandom sequence.

Example 5.65. Let $n : \mathbb{N} \to \mathbb{N}$ be an arbitrary function with $n(k) > k$ for all $k \in \mathbb{N}$. Define G by

$$G_{n(k)}(a_0 a_1 \cdots a_{k-1}) := \langle a_0 \cdots a_{\ell-1}, a_\ell \cdots a_{2\ell-1} \rangle_{i=0,\ldots,n(k)-1}, \quad \ell := \left\lfloor \frac{k}{2} \right\rfloor,$$

the PRBG corresponding to LFSR (cf. Example 5.53). For P as in Example 5.63 we obtain $\mu_{A_P \circ G \circ U_k}(1) - \mu_{A_P \circ U_{n(k)}}(1) \equiv -\frac{1}{2}$, not negligible. Hence, LFSR are *not* cryptographically secure. Note that this holds even for $n : k \mapsto k+1$: We have only 2^k possible pseudorandom strings of length $k+1$, in contrast to all $2 \cdot 2^k = 2^{k+1}$ possible random strings of length $k+1$. □

We state without proof:

Theorem 5.66 (Yao)**.** A PRBG is cryptographically secure iff it is **unpredictable**, i.e., iff it passes all polynomial next-bit tests. □

Empirical security

The problem with the LFSR is basically their linearity. Here are some attempts to destroy this linearity.

1. The first idea is to use the complete state vector $t^{(n)} = t^{(n-1)}C$ instead of simply returning its last entry $s_{n+\ell}$. For this use a non-linear "**filter function**" $f : \mathbb{F}_2^{1\times\ell} \to \mathbb{F}_2$, which will be regarded as part of the secret key:

[22] German: vernachlässigbar

Example 5.67. The following is known as Knapsack[23] generator: Given a primitive ℓ-bit LFSR (i.e., with period $2^\ell - 1$), fix a natural number $k > \lg \ell$ and choose in some random way non-negative integers $a_0, \ldots, a_{\ell-1}$. Together with the initial vector they build the secret key. Define the filter function $f(u) := (k\text{-th last bit of } \sum_{u_i=1} a_i)$ where $u = (u_0 \ldots u_{\ell-1}) \in \mathbb{F}_2^{1\times\ell}$ represents the current state vector. □

2. The second idea is to combine several LFSR **clocked**[24] in different ways:

   Example 5.68 (Alternating step generator). Let R be an LFSR generating a sequence $r = (r_n)$, and let S and S' be two different LFSR. Use R to reclock S and S' by setting

   $$\begin{array}{ll} s_{\ell+i} := t^{(i)} \cdot c \text{ and } s'_{\ell+i} := s'_{\ell+i-1}, & \text{if} \quad r_i = 0, \\ s'_{\ell+i} := t'^{(i)} \cdot c' \text{ and } s_{\ell+i} := s_{\ell+i-1}, & \text{if} \quad r_i = 1. \end{array}$$

   Define the resulting sequence to be $(s_i + s'_i)$ (in the notation of Definition 5.3). □

3. The third idea is that an LFSR throws away parts of another LFSR:

   Example 5.69 (Shrinking generator). Two LFSR are running in parallel and produce the bit sequences s and s'. If $s'_i = 1$ the bit s_i is returned, otherwise it is discarded[25]. □

Provable security

Let $f : \{0,1\}^\bullet \to \{0,1\}^\bullet$ be a polynomial deterministic algorithm. The question of whether there exists a polynomial algorithm that computes preimages of f leads us to the next fundamental definition:

Definition 5.70. Let $f : \{0,1\}^\bullet \to \{0,1\}^\bullet$ be a polynomial deterministic algorithm.

1. For an arbitrary polynomial probabilistic algorithm $A : \{0,1\}^\bullet \to \{0,1\}^\bullet$ define

   $$A_f : \{0,1\}^\bullet \to \{0,1\}, \quad x \mapsto \begin{cases} 1 & \text{if } A(f(x)) \in f^{-1}(f(x)) \\ 0 & \text{otherwise} \end{cases}.$$

   f is called a **one-way function (OWF)**[26] if $k \mapsto \mu_{A_f \circ U_k}(1)$ is negligible for all A.

2. Let $b : \{0,1\}^\bullet \to \{0,1\}$ be a polynomial deterministic algorithm. For an arbitrary polynomial statistical test (i.e., a polynomial probabilistic algorithm) $A : \{0,1\}^\bullet \to \{0,1\}$ define

   $$A_{f,b} : \{0,1\}^\bullet \to \{0,1\}, \quad x \mapsto \begin{cases} 1 & \text{if } A(f(x)) = b(x) \\ 0 & \text{otherwise} \end{cases}.$$

   b is called a **hardcore predicate**[27] (or **hardcore bit**, or **hidden bit**) **of** f if $k \mapsto \mu_{A_{f,b} \circ U_k}(1) - \frac{1}{2}$ is negligible for all A.

□

[23] German: Rucksack

[24] German: getaktet

[25] German: verworfen

[26] German: Einwegfunktion

[27] German: Hardcore-Prädikat, oder Sicherheitsbit

This means that a OWF f is easy to compute (polynomial deterministic) but hard to invert. A hardcore predicate b of a function f is a function that outputs a bit: If f is a OWF, then upon input $f(x)$ it is infeasible to correctly guess $b(x)$ with a non-negligible knowledge/advantage above $\frac{1}{2}$. Clearly, one can always guess the bit $b(x)$ with probability $\frac{1}{2}$. The hardcore predicate tries to explain in a concentrated sense the hardness of inverting the function f.

Remark 5.71.

1. If f is injective (in other words, does not lose information) and has a hardcore predicate then f is a OWF.
2. The existence of a hardcore predicate does not imply the injectivity of f. For example, the non-injective function f defined by $f(s_0 s_1 \cdots s_n) = s_1 \cdots s_n$ has the hardcore predicate $b(s_0 s_1 \cdots s_n) = s_0$.

□

Definition 5.72. A **one-way permutation (OWP)** is a bijective one-way function which is length preserving, i.e., $f(\{0,1\}^n) \subset \{0,1\}^n$ and $f : \{0,1\}^n \to \{0,1\}^n$ is a permutation for all $n \in \mathbb{N}_{>0}$. □

Theorem 5.73. Let f be a OWP with hardcore predicate b and $n : \mathbb{N} \to \mathbb{N}$ an arbitrary function with $n(k) > k$ which is bounded by some polynomial and which is computable by a polynomial run-time algorithm. Then the function $G : \{0,1\}^\bullet \to \{0,1\}^\bullet$ defined by

$$G_{n(k)}(s) := b(s)b(f(s)) \cdots b(f^{n(k)-1}(s))$$

is a CSPRBG with stretch function n. □

Proof. Consider

$$G'_{n(k)}(s) := b(f^{n(k)-1}(s)) \cdots b(f(s))b(s).$$

Assume G' is not cryptographically secure. Then Yao's Theorem would imply the existence of a next-bit test A_P which G' does not pass. But this contradicts b being a hardcore bit of f. The proof is complete since cryptographic security does not depend on the order of the output. ■

Lemma 5.74. Let f be a OWP. Then

$$g : \{0,1\}^{2n} \to \{0,1\}^{2n}, \ (x,y) \mapsto (f(x),y)$$

with $|x| = |y|$ defines a OWP with the **Goldreich-Levin hardcore predicate** b given by $b(x,y) := \sum_{i=1}^{n} x_i y_i \in \mathbb{F}_2$. □

Proof. This is a corollary of the Goldreich-Levin Theorem, see [Tre05]. ■

Corollary 5.75. The existence of a CSPRBG is equivalent to the existence of a OWP. □

Proof. The backward implication follows from Theorem 5.73 and Lemma 5.74. The forward implication is an exercise. ■

Remark 5.76. There is also the concept of a *pseudoramdom function family*. It is a family of efficiently computable functions that have the following property: No efficient algorithm can distinguish with a significant advantage between a function chosen from the family and a so-called *random oracle*. In practice often block ciphers are used where pseudorandom functions (for only a limited number of input and key sizes) are needed. In general, there is the concept by Goldreich–Goldwasser–Micaii to construct pseudorandom functions from pseudorandom generators. □

A CSPRBG based cryptosystem

We finish this chapter by constructing a cryptosystem based on a (public) CSPRBG G with stretch function n.

Example 5.77. Define the *symmetric* cryptosystem $(\mathscr{P}, \mathscr{C}, \mathscr{K}, \mathscr{E}, \mathscr{D})$ with $\mathscr{P} = \mathscr{C} = \{0,1\}^\bullet$, $\mathscr{K} = \{0,1\}^k$ for some **security parameter** $k := \lceil \lg |\mathscr{K}| \rceil \in \mathbb{N}$ (e.g., $k = 128$), and $\mathscr{E}$ as follows:
For each $p \in \mathscr{P}$ choose randomly a key $e \in \mathscr{K}$ and a seed $s \in \mathscr{K}$ and compute $G(s) \in \{0,1\}^{n(k)}$. Set

$$c = \mathscr{E}_e(p) := (s + e) \cdot (p + G(s))_{0,\dots,|p|-1},$$

where $+$ is the bitwise addition and $\cdot$ the concatenation of bits. So $|c|$ is slightly bigger than $|p|$. If $|p| > n(k)$ then choose several random seeds. This cryptosystem has at least two advantages:

1. After receiving $s + e$ one can compute s and start the computation of $G(s)$.

2. The receiver can decrypt c bitwise.

□

Chapter 6

Modern Symmetric Block Ciphers

In Chapter 4 we have seen that in order to get perfectly secret cryptosystems Shannon's theory states what is to do: It boils down to the idea of Vernam's one-time-pad, see Example 4.23. Clearly, in practice one could not apply these conditions. But we have seen in Remark 4.51 that there are several ideas to increase the unicity distance. Two main concepts that build the basis for modern symmetric block ciphers are confusion and diffusion:

Definition 6.1. Let Π be a cryptosystem.

1. **Confusion** is the concept that a digit resp. bit of the ciphertext $c \in \mathscr{C}$ depends on several parts of the key $e \in \mathscr{K}$.
2. **Diffusion** is the concept that if we change a digit resp. bit of the plaintext $p \in \mathscr{P}$ then the impact on the ciphertext $c \in \mathscr{C}$ is unpredictable. Statistically speaking: Changing *one bit of* p should change *half of the bits of* c. In the same way: If we change one bit of c then half of the bits of p should change.

□

These concepts are the underlying ideas of the well-known symmetric block ciphers DES and AES we discuss in the following. As a building block we first have a look at the so-called Feistel cipher.

6.1 Feistel cipher

The following ideas go back to Horst Feistel who worked at IBM. The cipher was first seen commercially in IBM's **Lucifer cipher** designed by Feistel and Don Coppersmith.

Let $\Sigma = \mathbb{Z}/2\mathbb{Z} = \mathbb{F}_2 = \{0, 1\}$ be an alphabet. The **Feistel cipher** is a block cipher of block length $2n$ for $n \in \mathbb{N}_{>0}$, so $\mathscr{P} = \mathscr{C} = \Sigma^{2n}$. One fixes some key space $\mathscr{K}$, a number of rounds $r \in \mathbb{N}_{>0}$ as well as a method on how to construct from one key $k \in \mathscr{K}$ a tuple $(k_0, \ldots, k_{r-1})$ of **sub-keys** for each round. For each sub-key k_i we denote by $f_{k_i} : \Sigma^n \to \Sigma^n$ the **round function** of the Feistel cipher. Let $p \in \mathscr{P}$ we subdivide $p = (L_0, R_0)$ such that $L_0, R_0 \in \Sigma^n$. In each round $0 \leq i < r$ one computes

$$(L_{i+1}, R_{i+1}) = (R_i, L_i \oplus f_{k_i}(R_i)).$$

The encryption function of the Feistel cipher is then given by

$$\mathscr{E}_k(p) = \mathscr{E}_k((L_0, R_0)) = (R_r, L_r) = c.$$

For decryption one just reverts the above process: Take a ciphertext $c = (R_r, L_r)$ then compute

$$(R_i, L_i) = (L_{i+1}, R_{i+1} \oplus f_{k_i}(L_{i+1})) \text{ for all } r > i \geq 0.$$

The decryption function of the Feistel cipher is then given by

$$\mathscr{D}_k(c) = \mathscr{D}_k((R_r, L_r)) = (L_0, R_0) = p.$$

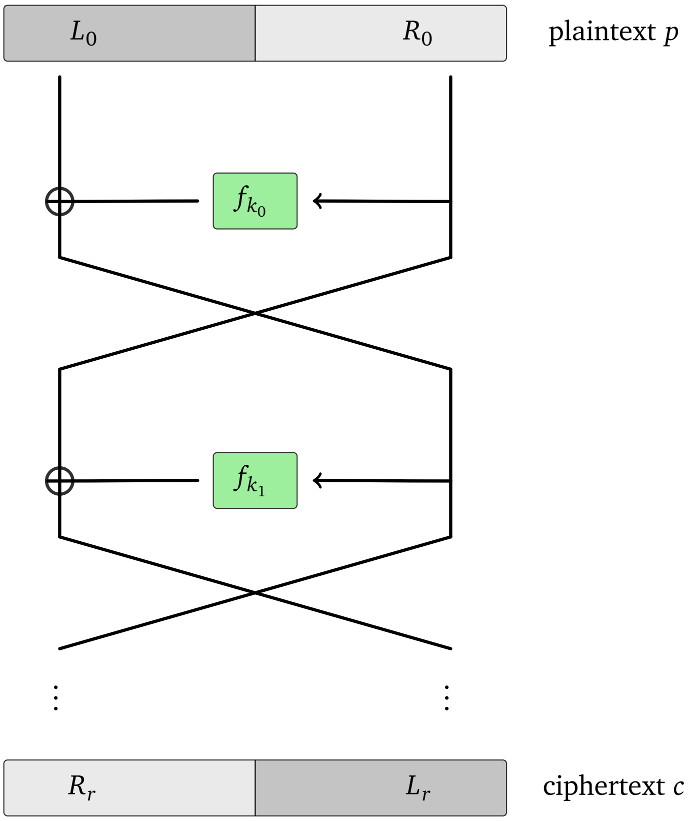


Figure 6.1: Encryption with the Feistel cipher

Remark 6.2. It is proven that if the round function f_{k_i} is a pseudorandom function (see Remark 5.76) then 3 rounds are sufficient to make the block cipher a pseudorandom permutation. □

6.2 Data Encryption Standard (DES)

DES stands for **Data Encryption Standard**, it is a symmetric block cipher created in 1972.

It is a modified variant of the Feistel cipher. Let $\Sigma = \{0, 1\}$ and the block length be 64, i.e. $\mathscr{P} = \mathscr{C} = \Sigma^{64}$. Moreover, $\mathscr{K} \subset \Sigma^{64}$ is defined by[1]

$$\mathscr{K} = \left\{ (b_1, \ldots, b_{64}) \in \Sigma^{64} \;\middle|\; \sum_{i=1}^{8} b_{8k+i} \equiv 1 \mod 2 \text{ for } 0 \leq k \leq 7 \right\}.$$

[1] So for each byte (8 bits) of one key the last bit in each byte is set such that the cross total is odd.

This means that the first 7 bits of a byte uniquely define the 8th bit. This is an encoding enabling corrections for transmission errors. It follows that a key really has only 56 bits, thus $|\mathscr{K}| = 2^{56} \approx 7.2 \cdot 10^{16}$.

DES has an **initial permutation** IP and a **final permutation** FP. Both permutations are inverse to each other ($\mathsf{FP} = \mathsf{IP}^{-1}$) and they do not have any cryptographic relevance. It was only used for efficient loading and writing of blocks on 1970s 8 bit hardware: Let $p = (p_1, \ldots, p_{64}) \in \mathscr{P}$ then

$$\mathsf{IP}(p_1, \ldots, p_{64}) = (p_{58}, p_{50}, \ldots, p_{15}, p_7).$$

The interesting step is the application of a Feistel cipher with 16 rounds (see illustration of one round in Figure 6.2): Starting with $p = (L_0, R_0) \in \mathscr{P}$ we apply the round function f_{k_i} where $k_i \in \Sigma^{48}$ is a subkey of $k \in \mathscr{K}$ for $0 \leq i < 16$.

1. By construction of the Feistel cipher, f_{k_i} is applied to one half of the text block, i.e. on 32 bits. To these 32 input bits, say R, an **expansion function** E is applied: $E : \Sigma^{32} \to \Sigma^{48}$, $E(R) \in \Sigma^{48}$ is just R with bits permuted and half of its bits duplicated.

2. Next one applies the subkey k_i to $E(R)$ using XOR:

$$E(R) \oplus k_i = (B_1, \ldots, B_8) \in \Sigma^{48}.$$

   Here, $B_j \in \Sigma^6$ for $1 \leq j \leq 8$.

3. Next, to each B_j one applies *non-linear* functions $S_j : \Sigma^6 \to \Sigma^4$ that are provided by lookup tables. These S_j are called **S-boxes**.

$$S_j(B_j) = S_j(b_{j,1}, \ldots, b_{j,6}) = (c_{j,1}, \ldots, c_{j,4}) = C_j \in \Sigma^4.$$

4. Finally one applies a permutation function $P : \Sigma^{32} \to \Sigma^{32}$ on C. We denote $P(C)$ then as the result of $f_{k_i}(R)$.

In short:

$$p \Rightarrow \mathsf{IP}(p) = (L_0, R_0) \overset{\text{Feistel}}{\Longrightarrow} (R_{16}, L_{16}) \Rightarrow \mathsf{FP}(R_{16}, L_{16}) = c.$$

This describes DES' encryption function. In order to decrypt, one just applies the encryption function with inverted key order.

Remark 6.3.

1. Note that the S-boxes provide the core security of DES: All other steps are linear and we have already seen in Section 3.4 that linearity is trivially breakable nowadays.

2. DES is considered insecure nowadays, mainly due to its short key size of 56 bits. The first practical attack was done in 1999: Researcher broke a DES key in under 23 hours by a brute-force attack ($2^{56} \approx 7.2 \cdot 10^{16}$ possible keys), i.e. trying every possible key. There are some theoretical attacks on DES, still **Triple DES** is believed to be practically secure. The security of Triple DES comes from the fact that the encryption functions do not build a group, otherwise one could find for any two keys $k_1, k_2 \in \mathscr{K}$ another key $k_3 \in \mathscr{K}$ such that $\mathscr{E}_{k_2} \circ \mathscr{E}_{k_1} = \mathscr{E}_{k_3}$ which would make the application of several keys cryptographically useless.

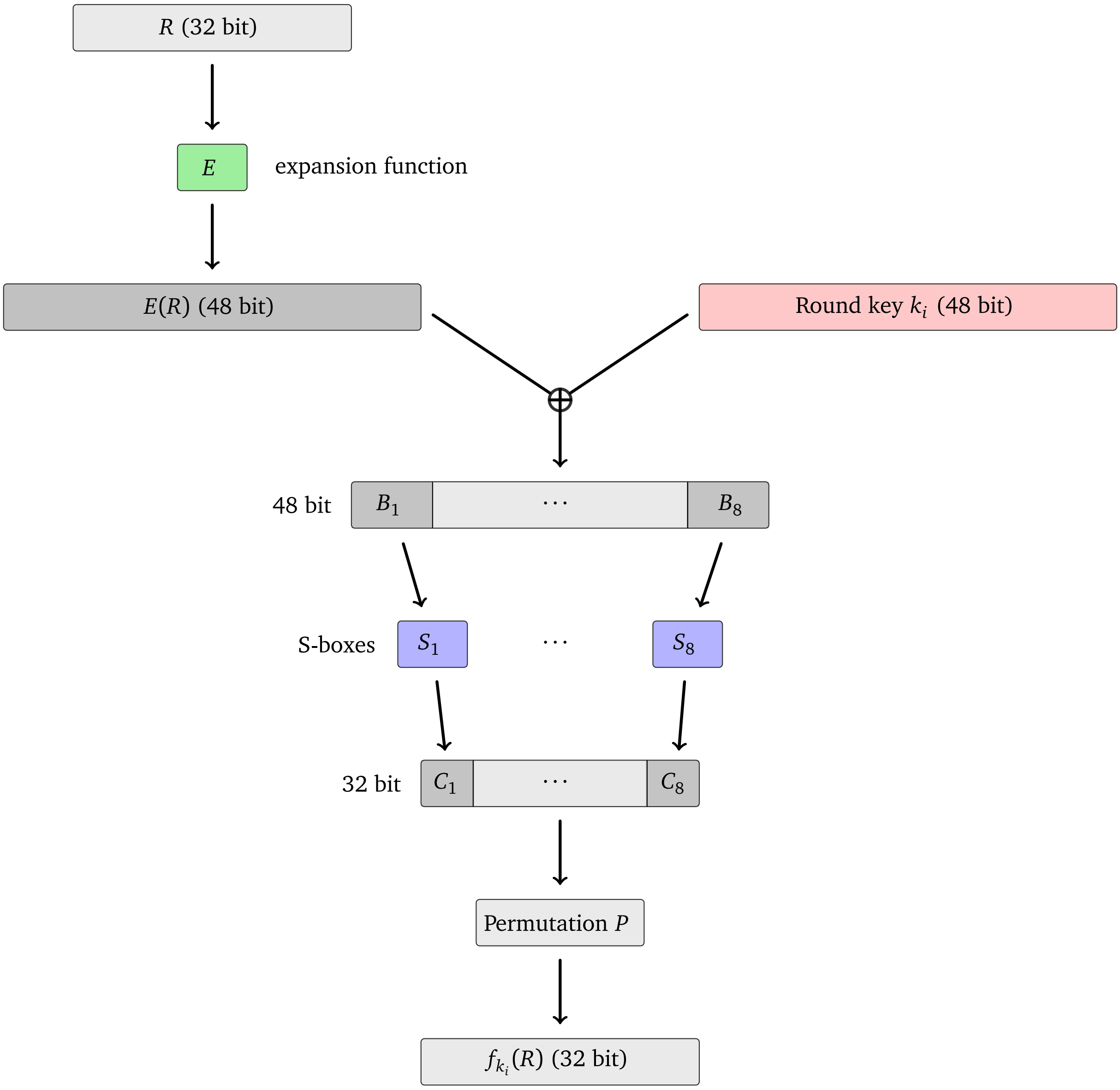


Figure 6.2: One round of the Feistel function in DES

3. The NSA took part in the process of standardizing DES. There are (not proven!) suspicions that the NSA was one of the parties that suggested to reduce the key size from 128 bits to 64 bits (resp. 56 bits). This decision was mainly made due to practical reasons: In the 1970s 56 bits fit on a single chip, whereas 128 bits did not. Still, people believe that the NSA had already in the 1970s enough computing power to do brute force attacks on DES.

□

6.3 Advanced Encryption Standard (AES)

AES is a subset resp. specialized instantiation of the **Rijndael cipher** named after two Belgian cryptographers, Joan Daemen and Vincent Rijmen. AES was standardized by the U.S. NIST[2] after a process of 5 years. It was announced on 26th of November 2001.

AES, as DES, is based on substitution and permutation and is fast in software and hardware.[3] In contrast to DES, AES does not use the Feistel cipher. Let us assume the following general concepts for a block cipher:

Let $\Sigma = F = \mathbb{F}_{2^n} = \mathbb{F}_2/\langle f\rangle$ be the finite field with 2^n elements. We list four different actions on $B := F^\ell$.

1. `SubByte` or `S-box`: The inversion in the field F defines a permutation $^{-1} : a \mapsto a^{-1}$ for $a \in F^*$ and $0^{-1} := 0$. This permutation is *non-linear* but fixes $0, \pm 1$. Choose an $\mathbb{F}_2$-linear invertible map $g : F \to F$ and an element $t \in F$ such that $\sigma : F \to F,\ a \mapsto ga^{-1} + t$ is a *fixed-point-free* permutation (or **derangement**). Extend σ to a permutation $p = (a_1, \dots, a_\ell) \mapsto (\sigma(a_1), \dots, \sigma(a_\ell))$ on B.

2. `ShiftRows`: A permutation $\pi \in S_\ell$ induces a *block permutation* on B defined by
$$(a_1, \dots, a_\ell) \mapsto \left(a_{\pi(1)}, \dots, a_{\pi(\ell)}\right).$$

3. `MixColumns`: Choose an element $h \in F[x]$ of degree $m \mid \ell$ and an invertible element c in the residue class ring[4] $R := F[x]/\langle h\rangle$. Then $c \in R^*$ induces a permutation $c : F^m \to F^m, p \mapsto c \cdot p$, where $p = (a_1, \dots, a_m)$ is identified with the polynomial $a_m x^{m-1} + \cdots + a_2 x + a_1$. Extend this permutation to a permutation on $B = F^\ell = (F^m)^{\frac{\ell}{m}}$ by $p = (p_1, \dots, p_{\frac{\ell}{m}}) \mapsto (c \cdot p_1, \dots, c \cdot p_{\frac{\ell}{m}})$.

4. `AddRoundKey`: In case $\mathcal{K} = B$ then the addition of a key e induces a permutation $p \mapsto p + e$ on $B = F^\ell$.

Note that (1) and (2) commute but (1) and (3) don't.

The specific instantiation of AES now looks as follows:

1. $n = 8$: The 256 elements in the field $F = \mathbb{F}_{2^8} = \mathbb{F}_{256}$ are considered as bytes (= 8 bits) and represented by two hexadecimal digits $\mathtt{00}, \mathtt{01}, \dots, \mathtt{0F}, \mathtt{10}, \dots, \mathtt{FF}$. As customary we write `0x` in front of hexadecimal numbers. So `0x63` is the hexadecimal representation of the decimal number $99 = 6 \cdot 16^1 + 3 \cdot 16^0$. Its binary representation is 01100011.

2. $\ell := 16$: $B = F^{16} \cong_{\mathbb{F}_2} \mathbb{F}_2^{128}$ which has more elements than atoms in the universe.

3. $f := f_{\text{AES}} := x^8 + x^4 + x^3 + x + 1$: Then `0x63` corresponds to the field element $\bar{x}^6 + \bar{x}^5 + \bar{x} + 1 \in F$.

[2] National Institute of Standards and Technology

[3] Modern CPUs have AES functionality in hardware, e.g. for data volume encryption.

[4] Note that we do not require R to be a field.

4. $t := \texttt{0x63} \in F$ corresponding to the vector

$$t := \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \text{ and choose } g := \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \in \mathrm{GL}_8(\mathbb{F}_2)$$

For the lookup table of the permutation $F \to F,\ a \mapsto ga^{-1} + t$ see Figure 6.3.[5]

```
   | 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
00 |63 7c 77 7b f2 6b 6f c5 30 01 67 2b fe d7 ab 76
10 |ca 82 c9 7d fa 59 47 f0 ad d4 a2 af 9c a4 72 c0
20 |b7 fd 93 26 36 3f f7 cc 34 a5 e5 f1 71 d8 31 15
30 |04 c7 23 c3 18 96 05 9a 07 12 80 e2 eb 27 b2 75
40 |09 83 2c 1a 1b 6e 5a a0 52 3b d6 b3 29 e3 2f 84
50 |53 d1 00 ed 20 fc b1 5b 6a cb be 39 4a 4c 58 cf
60 |d0 ef aa fb 43 4d 33 85 45 f9 02 7f 50 3c 9f a8
70 |51 a3 40 8f 92 9d 38 f5 bc b6 da 21 10 ff f3 d2
80 |cd 0c 13 ec 5f 97 44 17 c4 a7 7e 3d 64 5d 19 73
90 |60 81 4f dc 22 2a 90 88 46 ee b8 14 de 5e 0b db
a0 |e0 32 3a 0a 49 06 24 5c c2 d3 ac 62 91 95 e4 79
b0 |e7 c8 37 6d 8d d5 4e a9 6c 56 f4 ea 65 7a ae 08
c0 |ba 78 25 2e 1c a6 b4 c6 e8 dd 74 1f 4b bd 8b 8a
d0 |70 3e b5 66 48 03 f6 0e 61 35 57 b9 86 c1 1d 9e
e0 |e1 f8 98 11 69 d9 8e 94 9b 1e 87 e9 ce 55 28 df
f0 |8c a1 89 0d bf e6 42 68 41 99 2d 0f b0 54 bb 16
```

Figure 6.3: Lookup table for the Rijndael S-box

5. π is the permutation inducing the following row-shifts

$$p := \begin{pmatrix} a_1 & a_5 & a_9 & \boxed{a_{13}} \\ a_2 & a_6 & a_{10} & \boxed{a_{14}} \\ a_3 & a_7 & a_{11} & \boxed{a_{15}} \\ a_4 & a_8 & a_{12} & \boxed{a_{16}} \end{pmatrix} \mapsto \begin{pmatrix} a_1 & a_5 & a_9 & \boxed{a_{13}} \\ a_6 & a_{10} & \boxed{a_{14}} & a_2 \\ a_{11} & \boxed{a_{15}} & a_3 & a_7 \\ \boxed{a_{16}} & a_4 & a_8 & a_{12} \end{pmatrix} \in B = F^{16} \equiv F^{4 \times 4}.$$

6. $m = 4$, $h := x^4 + 1 = (x+1)^4 \in F[x]$, and $c = \texttt{0x03} \cdot x^3 + x^2 + x + \texttt{0x02} \in R^*$. This corresponds

[5] The entries are hexadecimal numbers where we dropped the 0x-prefix.

to the matrix[6] multiplication

$$p := \begin{pmatrix} a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \\ a_4 & a_8 & a_{12} & a_{16} \end{pmatrix} \mapsto \underbrace{\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}}_{\in F^{4\times 4}} \cdot p \in F^{4\times 4} \equiv F^{16} =: B$$

7. $\mathcal{K} \in \left\{ \underbrace{B = F^{16}}_{\text{128-bit keys}}, \underbrace{F^{24}}_{\text{192-bit keys}}, \underbrace{F^{32}}_{\text{256-bit keys}} \right\}.$

As DES, AES is round-based, depending on $\mathcal{K} = F^{16}, F^{24}, F^{32}$ we have $r = 10, 12, 14$ and in total $r+1$ rounds. The so-called **Rijndael key schedule** produces the $r+1$ 128 bit round key blocks for each round. In the $(r+1)$st round AES computes only

1. `SubBytes`,
2. `ShiftRows` and
3. `AddRoundKey`

but it does not compute `MixColumns` as done in all r rounds beforehand.

Remark 6.4.

1. The iteration of the two steps `ShiftRows` and `MixColumns` produce diffusion: Changing a bit in p would change each bit in $c = \mathcal{E}_e(p)$ with probability $\frac{1}{2}$.
2. Clearly, AES is in the near future secure against brute force attacks: $2^{128} \approx 3.4 \cdot 10^{38}$ keys have to be checked. There are various other attacks against AES, still none of these is practical at the moment and AES is believed to be secure nowadays.

□

[6] The entries are hexadecimal numbers where we dropped the `0x`-prefix.

Chapter 7

Candidates of One-Way Functions

In order to construct asymmetric crypto systems we need to investigate the concept of one-way functions (see Chapter 5) in more detail. First, let us recall the main complexity assumptions we need for proving the existence of cryptographically useful one-way functions.

7.1 Complexity classes

Let us recall the main statements from our introduction to complexity theory in Section 2.1.

Definition 7.1 (see Definition 2.7)**.** A problem instance P lies in the complexity class

1. P if P is solvable by a deterministic algorithm with polynomial runtime.
2. BPP if P is solvable by a probabilistic algorithm with polynomial runtime.
3. BQP if P is solvable by a deterministic algorithm on a quantum computer in polynomial runtime.
4. NP if P is verifiable by a deterministic algorithm with polynomial runtime.
5. NPC if any other problem in NP can be reduced resp. transformed to P in polynomial time.
6. EXP if P is solvable by a deterministic algorithm with exponential runtime.

□

Definition 7.2. Let $G = \langle g \rangle$ be a finite cyclic group. For each $y \in G$, there is exactly one minimal $a \in \mathbb{N}_0$ with $g^a = y$. We call a the **discrete logarithm** of y with **basis** g. Computing $a =$"$\log_g y$" (given g and y) is called the **discrete logarithm problem (DLP).** □

Remark 7.3. In modern cryptography we make the following two standard assumptions:

1. **DL assumption**: DLP $\notin$ BPP, i.e., the computation of the **discrete logarithm** in the group $\mathbb{F}_p^*$ is not in BPP.
2. **FACTORING assumption**: The factorization of natural numbers does not lie in BPP.

We can prove the existence of cryptographically useful one-way functions *only under such assumptions*. □

Example 7.4. For each prime p choose a fixed primitive element $a \in \mathbb{Z}/p\mathbb{Z}$. Assuming DL, the function

$$f : \{1, \dots, p-1\} \to \{1, \dots, p-1\}, x \mapsto a^x \bmod p.$$

is a OWF with hardcore predicate

$$b(x) = \begin{cases} 1 & \text{if } x < \frac{p}{2}, \\ 0 & \text{if } x \geq \frac{p}{2}. \end{cases}$$

□

7.2 Squaring modulo n

Consider the squaring homomorphism

$$q_n : (\mathbb{Z}/n\mathbb{Z})^* \to (\mathbb{Z}/n\mathbb{Z})^*, x \mapsto x^2.$$

Remark 7.5.

1. If $n = p$ is a *prime* then
   a) $\ker q_n = \{\pm 1\}$.
   b) If $p > 2$ then there are exactly $\frac{p-1}{2}$ squares in $(\mathbb{Z}/p\mathbb{Z})^*$.
2. If $n = pq$, a product of two distinct odd primes p, q. Then
   a) $\ker q_n$ consists of four elements.
   b) There exists exactly $\frac{\varphi(n)}{4}$ squares in $(\mathbb{Z}/n\mathbb{Z})^*$.

□

Example 7.6. Consider the following values of n:

1. $n = 3$:

a	1	2
a^2	1	1

2. $n = 5$:

a	1	2	3	4
a^2	1	4	4	1

3. $n = 15$:

a	1	2	4	7	8	11	13	14
a^2	1	4	1	4	4	1	4	1

Note that $\varphi(15) = \varphi(3)\varphi(5) = 2 \cdot 4 = 8$, and thus, $\frac{\varphi(15)}{4} = 2$.

□

Now we want to study classical methods to identify squares and compute square roots.

7.3 Quadratic residues

Definition 7.7. For $a \in \mathbb{Z}$ and p prime define the **Legendre symbol**

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & \text{if } a \equiv 0 \bmod p \text{ (i.e. } p \mid a\text{)}, \\ 1 & \text{if } a \not\equiv 0 \bmod p \text{ and } a \equiv b^2 \bmod p \text{ (for some } b \in \mathbb{Z}\text{)}, \\ -1 & \text{otherwise.} \end{cases}$$

□

Theorem 7.8 (Euler)**.** Let $p > 2$ be an odd prime. Then $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \bmod p$ □

Proof. The case $p \mid a$ is clear. So assume $p \nmid a$. The group $\mathbb{F}_p^* = (\mathbb{Z}/p\mathbb{Z})^*$ is cyclic of order $p-1$ so $a^{p-1} \equiv 1 \bmod p$. Hence $a^{\frac{p-1}{2}}$ is a root of $x^2 - 1 \in \mathbb{F}_p[x]$ and the group homomorphism

$$h : (\mathbb{Z}/p\mathbb{Z})^* \to \{\pm 1\} \leq (\mathbb{Z}/p\mathbb{Z})^*, \ a \mapsto a^{\frac{p-1}{2}}$$

is surjective. The kernel of h thus consists of $\frac{p-1}{2}$ elements and contains $((\mathbb{Z}/p\mathbb{Z})^*)^2$, so it coincides with $((\mathbb{Z}/p\mathbb{Z})^*)^2$. ■

Euler's theorem can be used to simplify the computation of the Legendre symbol. For example

Corollary 7.9.

$$\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}} = \begin{cases} 1 & \text{if} \quad p \equiv 1 \bmod 4 \\ -1 & \text{if} \quad p \equiv 3 \bmod 4 \end{cases}$$

□

Exercise 7.10. The map

$$\left(\frac{\cdot}{p}\right) : (\mathbb{Z}/p\mathbb{Z})^* \to \{\pm 1\}$$

is a group homomorphism. □

Definition 7.11. The elements in the kernel of $\left(\frac{\cdot}{p}\right)$ are called **quadratic residues** modulo p. □

Definition 7.12 (Jacobi symbol)**.** Let $n = p_1^{a_1} \cdots p_r^{a_r} > 1$ be the decomposition of the natural number n as powers of distinct primes. For $a \in \mathbb{Z}$ set

$$\left(\frac{a}{n}\right) := \left(\frac{a}{p_1}\right)^{a_1} \cdots \left(\frac{a}{p_r}\right)^{a_r} \in \{-1, 0, 1\},$$

and for $n = 1$ set $\left(\frac{a}{1}\right) := 1$. □

The Jacobi symbol can be computed without knowing an explicit factorization of n (see lecture on elementary number theory (EZT)).

Corollary 7.13. $\left(\frac{a}{n}\right) = -1$ implies that a is not a square modulo n. □

Definition 7.14. If $\left(\frac{a}{n}\right) = 1$ and a is not a square modulo n, then we call a a **pseudo-square modulo** n. □

Example 7.15. Recall Example 7.6 and consider the following values of n:

1. $n = 3$:

a	1	2
a^2	1	1
$\left(\frac{a}{n}\right)$	1	-1

2. $n = 5$:

a	1	2	3	4
a^2	1	4	4	1
$\left(\frac{a}{n}\right)$	1	-1	-1	1

3. $n = 15$:

a	1	2	4	7	8	11	13	14
a^2	1	4	1	4	4	1	4	1
$\left(\frac{a}{n}\right)$	1	1	1	-1	1	-1	-1	-1

So 2 and 8 are pseudo-squares modulo 15.

□

Definition 7.16. Define the following sets:

1. The squares modulo n:

$$Q_n := \left\{a \in (\mathbb{Z}/n\mathbb{Z})^* \mid \exists b \in \mathbb{Z}/n\mathbb{Z} : a = b^2\right\} = ((\mathbb{Z}/n\mathbb{Z})^*)^2.$$

2. The non-squares modulo n:

$$\overline{Q}_n := \left\{a \in (\mathbb{Z}/n\mathbb{Z})^* \mid \nexists b \in \mathbb{Z}/n\mathbb{Z} : a = b^2\right\} = (\mathbb{Z}/n\mathbb{Z})^* \setminus ((\mathbb{Z}/n\mathbb{Z})^*)^2.$$

3. The pseudo-squares modulo n:

$$\widetilde{Q}_n := \left\{a \in (\mathbb{Z}/n\mathbb{Z})^* \mid \left(\frac{a}{n}\right) = 1\right\} \setminus Q_n \subset \overline{Q}_n.$$

□

Proposition 7.17. Let p be a prime with $p \equiv 3 \bmod 4$. Then $a \in Q_p$ has exactly one square root in Q_p. We call it the **principal root of** a. □

Proof. $\mathbb{Z}/p\mathbb{Z} \cong \mathbb{F}_p$ is a field, hence there are exactly two roots $\pm b$ for a (i.e. $a = b^2$). By Corollary 7.9 and Exercise 7.10 we compute $\left(\frac{-b}{p}\right) = \left(\frac{-1}{p}\right)\left(\frac{b}{p}\right) = (-1)^{\frac{p-1}{2}}\left(\frac{b}{p}\right) = -\left(\frac{b}{p}\right)$ since $p \equiv 3 \bmod 4$ and thus $\frac{p-1}{2}$ is odd. W.l.o.g. we can assume that $\left(\frac{b}{p}\right) = 1$ and $\left(\frac{-b}{p}\right) = -1$. Thus we have $b \in Q_p$ and $-b \in \overline{Q}_p$. ■

Example 7.18.

1. Let $p = 5$, so $5 \equiv 1 \bmod 4$: $Q_5 = \{1,4\}$. The square roots of 4 are $2,3 \in \overline{Q}_5$. The square roots of 1 are $1,4 \in Q_5$.
2. Now let $p = 7$ with $7 \equiv 3 \bmod 4$: $Q_7 = \{1,2,4\}$. The square roots of 2 are $3 \in \overline{Q}_7$ and $4 \in Q_7$. The square roots of 4 are $2 \in Q_7$ and $5 \in \overline{Q}_7$.

□

Definition 7.19. An $n \in \mathbb{N}$ is called a **Blum number** if $n = p_1p_2$ with p_1, p_2 prime numbers, $p_1 \neq p_2$ and $p_i \equiv 3 \bmod 4$ for $i = 1,2$. □

Lemma 7.20. The following holds for a Blum number n:

1. Each $a \in Q_n$ has exactly one square root in Q_n (again called the **principal root of** a), one in $\widetilde{Q}_n$, and two in $\overline{Q}_n \setminus \widetilde{Q}_n$.
2. $-1 \in \widetilde{Q}_n$.

□

Proof.

1. Follows from Proposition 7.17 and the Chinese Remainder Theorem. Details are left as an exercise.
2. By Definition 7.12 we get $\left(\frac{-1}{n}\right) = \left(\frac{-1}{p_1}\right)\left(\frac{-1}{p_2}\right)$. Since $p_i \equiv 3 \bmod 4$ we can follow the statement by Corollary 7.9. ■

7.4 Square roots

Definition 7.21. We list the following three fundamental problems:

1. FACTORING : Given an $n \in \mathbb{N}$ compute a prime factor.
2. SQROOT : Given an $n \in \mathbb{N}$ and a square $a \in Q_n$ compute a square root of a modulo n.
3. QRP : Given an $n \in \mathbb{N}$ and $a \in \mathbb{Z}$ with $\left(\frac{a}{n}\right) = 1$ decide whether a is a square or a pseudosquare.

□

Theorem 7.22. SQROOT for $n = p$ prime lies in BPP. □

Proof. Let $n = p > 2$ a prime number and $a \in (\mathbb{Z}/p\mathbb{Z})^*$. The idea is to exploit that

$$a^m = 1 \text{ for } m \text{ odd implies that } \left(a^{\frac{m+1}{2}}\right)^2 = a^{m+1} = a. \tag{7.1}$$

Recall that

$$a \in Q_p \iff a^{\frac{p-1}{2}} \equiv 1 \bmod p,$$

by Euler's Theorem 7.8. So let a be a square.

1. $\frac{p-1}{2}$ is odd, i.e., $p \equiv 3 \bmod 4$:
   Using (7.1) with $m = \frac{p-1}{2}$ yields the square root $a^{\frac{m+1}{2}} = a^{\frac{p+1}{4}}$ of a.

2. $\frac{p-1}{2}$ is even, i.e., $p \equiv 1 \bmod 4$:

$$a \in Q_p \iff a^{\frac{p-1}{2}} = 1 \iff a^{\frac{p-1}{4}} = \pm 1.$$

   We now prove by induction that we can use the equation $a^m = \pm 1$ for $m \mid \frac{p-1}{4}$ to compute a square root of a. We start with $m = \frac{p-1}{4}$.

   a) $a^m = 1$ and m even: Proceed with the equation $a^{\frac{m}{2}} = \pm 1$.

   b) $a^m = 1$ and m odd: (7.1) yields a square root $a^{\frac{m+1}{2}}$ of a.

   c) $a^m = -1$. *Choose* an arbitrary $b \in \overline{Q}_p$ and set $b' := b^{\frac{p-1}{4m}}$. Proceed with the equation

$$(ab'^2)^m = a^m b^{\frac{p-1}{2}} = (-1)^2 = 1,$$

   since b is not a square and thus by Theorem 7.8 $\left(\frac{b}{p}\right) = b^{\frac{p-1}{2}} \equiv -1 \bmod p$. Finally note that if c is a square root of ab'^2 then $a = (cb'^{-1})^2$.

   This describes the probabilistic polynomial algorithm of **TonelliShanks** [Wik17j]. We omit the details. ■

Example 7.23. Let $p = 41$. We know that $a = -2 \in Q_{41}$ since $(-2)^{21} = -(2^7)^3 = -5^3 = -2$ so $(-2)^{20} = 1$. Now we want to use the above algorithm to compute a square root of a. Note that $\frac{p-1}{2} = 20$ is even and $\frac{p-1}{4} = 10$. Find an element $b \in \overline{Q}_{41}$ by randomly checking (probability of failure is $\frac{1}{2}$):

1. $2^{20} = (-1)^{20} \cdot (-2)^{20} = 1$ (×).
2. $3^{20} = (3^4)^5 = (81)^5 = (-1)^5 = -1$ (√).

So choose $b = 3$:

	a	m	a^m	$\frac{p-1}{4m}$	$b' = b^{\frac{p-1}{4m}}$	$\frac{m+1}{2}$	b'^{-1}		$\sqrt{a}$
	-2	10	-1	1	$3^1 = 3$		14	$33 \cdot 14 =$	11
$-2 \cdot 3^2 =$	23	10	1						
	23	5	-1	2	$3^2 = 9$		32	$10 \cdot 32 =$	33
$23 \cdot 9^2 = -23 =$	18	5	1			3		$18^3 =$	10

□

Lemma 7.24. Let $n = p_1 p_2$ be a Blum number ($p_i \equiv 3 \bmod 4$ will not be relevant). Any $x \in \ker q_n$ with $x \neq \pm 1$ yields a factorization of n. □

Proof. Let $x \in \{m \in \mathbb{N} \mid 1 < m < n-1 \text{ and } m^2 \equiv 1 \bmod n\}$. Then $p_1 p_2 = n \mid x^2 - 1 = (x-1)(x+1)$. Since $n \nmid x \pm 1$ we conclude w.l.o.g. that $p_1 \mid x-1$ and $p_2 \mid x+1$. Now p_1 can be effectively computed as $p_1 = \gcd(x-1, n)$. ■

Theorem 7.25. If SQROOT for Blum numbers lies in BPP then FACTORING for Blum numbers lies in BPP. □

Proof. From a probabilistic polynomial algorithm A that solves SQROOT for Blum numbers we construct the following probabilistic polynomial algorithm that solves FACTORING:
Choose an arbitrary element $c \in (\mathbb{Z}/n\mathbb{Z})^*$ and compute $a := A(c^2)$. So $\frac{c}{a}$ is an element in $\ker q_n$ which is with probability $\frac{1}{2}$ different from ± 1. The rest is Lemma 7.24 ■

7.5 One-way functions

We sharpen our previous assumption on factorization:

1. **FACTORING assumption**: FACTORING of Blum numbers does not lie in BPP.
2. **QR assumption**: QRP for Blum numbers does not lie in BPP.

Theorem 7.26. Let n be a Blum number. Then $f := q_{n|Q_n} : Q_n \to Q_n$ is a permutation.

1. f is a one-way permutation (OWP) under the FACTORING assumption (with security parameter: $k := \lceil \lg |Q_n| \rceil = \left\lceil \lg \frac{\varphi(n)}{4} \right\rceil = \lceil \lg \varphi(n) \rceil - 2$).
2. The so-called **parity (bit)**

$$\mathrm{par} : (\mathbb{Z}/n\mathbb{Z})^* \to \{0, 1\},\ a \mapsto (\text{smallest nonnegative representative of a}) \bmod 2$$

defines under the QR assumption a hardcore bit of f.

□

Proof. f is a permutation by Lemma 7.20 (1) since each $a \in Q_n$ has exactly one square root in Q_n.

1. From Theorem 7.25 Statement (1) follows.
2. To prove (2) let $\left(\frac{a}{n}\right) = 1$, i.e., $a \in Q_n \dot{\cup} \widetilde{Q}_n$. For the principal root $w \in Q_n$ of a^2 we claim:

$$w = a \iff \mathrm{par}(w) = \mathrm{par}(a). \tag{7.2}$$

The forward implication of the claim is trivial. We now prove the backward implication: Since $-1 \in \widetilde{Q}_n$ by Lemma 7.20 (2) and $w \in Q_n$ we deduce that $-w \in \widetilde{Q}_n$ (i.e., that $\widetilde{Q}_n = -Q_n$). So $a = w$ or $a = -w$. In other words: $a \neq w \implies a = -w \implies \mathrm{par}(w) \neq \mathrm{par}(a)$ (remember, n as a Blum number is odd). So the above claim holds. From an algorithm B for which $B(x)$ with $x = f(a) = a^2$ predicts $\mathrm{par}(a)$ we obtain an algorithm for QRP by returning:

$$\begin{cases} a \text{ is a square} & \text{if } B(a^2) = \mathrm{par}(w), \\ a \text{ is a pseudo-square} & \text{if } B(a^2) \neq \mathrm{par}(w). \end{cases}$$

Due to the QR assumption such an algorithm does not lie in BPP. ■

Definition 7.27. The function f is called the **Rabin function**. The PRBG G constructed according to Theorem 5.73 is called the **Blum-Blum-Shub generator** (see [Wik17a]): For a Blum number n and a **seed** $s \in (\mathbb{Z}/n\mathbb{Z})^*$ define $G(s) = x_0x_1x_2\dots$ with $x_i = \operatorname{par}(s^{2^i})$. G is then a CSPRBG under the QR assumption for Blum numbers. □

7.6 Trapdoors

A OWP $f : \{0,1\}^\bullet \to \{0,1\}^\bullet$ can be viewed as a family of permutations $f_k : \{0,1\}^k \to \{0,1\}^k$.

To define a OWP with a **trapdoor**[1] we need the following properties and structures:

1. I, an infinite **index set**,
2. $|\cdot| : I \to \mathbb{N}$, a **length function**.
3. $I_k := \{i \in I : |i| \leq k\}$ (we call k the **security parameter**).
4. X_i for all $i \in I$, a family of finite sets.
5. $f = (f_i)_{i\in I} : \bigcup_{i\in I} X_i \to \bigcup_{i\in I} X_i$, a family of permutations $f_i : X_i \to X_i$.
6. A **trapdoor information** t_i for all $i \in I$ (see the examples below).
7. $\mathscr{E}$, a polynomial algorithm with $\mathscr{E}(i,x) = \mathscr{E}_i(x) = f_i(x)$ for all $i \in I$ and $x \in X_i$.
8. $\mathscr{D}$, a polynomial algorithm with $\mathscr{D}(i,t_i,f_i(x)) = \mathscr{D}_{(i,t_i)}(f_i(x)) = x$ for all $i \in I$ and $x \in X_i$.
9. $S_k := \{(i,x) \mid i \in I_k, x \in X_i\}$, the possible inputs of $\mathscr{E}$ with security parameter k.

For a probabilistic algorithm $A : \bigcup_{i\in I} X_i \to \bigcup_{i\in I} X_i$ with output $A(i,y) \in X_i$ for all $i \in I$ and $y \in X_i$ define the probabilistic algorithm A_f by setting

$$A_f(i,x) = \begin{cases} 1 & \text{if } A(i,f_i(x)) = x, \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in I$ and $x \in X_i$.

As usual, let U_{S_k} denote the uniformly distributed random variable on S_k (i.e., first choose a random $i \in I$ and then a random $x \in X_i$). Then $\mu_{A_f \circ U_{S_k}}(1)$ is the probability of A correctly computing the preimage x of $y = f_i(x)$.

Definition 7.28. The permutation $f = (f_i)_{i\in I} : \bigcup_{i\in I} X_i \to \bigcup_{i\in I} X_i$ is called a **one-way permutation with trapdoor** $t = (t_i)_{i\in I}$ if $k \mapsto \mu_{A_f \circ U_{S_k}}(1)$ is negligible for all polynomial algorithms A as above (cf. Definition 5.70 (1)). □

Example 7.29 (Rabin function). Set $I :=$ {Blum numbers}, $|n| = k = \lfloor \lg n \rfloor$, $X_n = Q_n$, $f_n = q_{n|Q_n} : x \mapsto x^2 \bmod n$, t_n the factorization of n, and $\mathscr{D}$ is the combination of the algorithm in the proof of Theorem 7.22 (Tonelli-Shanks) with the Chinese Remainder Theorem. We obtain a one-way permutation with trapdoor under the FACTORING assumption, which is equivalent to the "SQROOT $\notin$ BPP" assumption for Blum numbers.[2] The parity bit $b := \operatorname{par}$ is a hardcore bit under the QR assumption (by Theorem 7.26 (2)). □

[1] German: wörtlich Falltür, man sagt aber im Deutschen Hintertür

[2] The equivalence is Theorem 7.22 combined with the Chinese Remainder Theorem and Theorem 7.25.

7.7 The Blum-Goldwasser construction

Given a OWP with trapdoor and hardcore bit b Blum and Goldwasser constructed the following asymmetric *probabilistic*[3] cryptosystem $(\mathscr{P}, \mathscr{C}, \kappa, \mathscr{E}, \mathscr{D})$ with

1. $\mathscr{P} = \mathscr{C} = \{0,1\}^\bullet$,
2. $\mathscr{K} = I$, $\mathscr{K}' = \{(i, t_i) \mid i \in I\}$, $\kappa : \mathscr{K}' \to \mathscr{K}, (i, t_i) \mapsto i$,
3. $\mathscr{E}_e : \mathscr{P} \rightsquigarrow \mathscr{C}$, $\mathscr{D}_d : \mathscr{C} \to \mathscr{P}$ as follows (compare with Example 5.77):
   Let $e \in \mathscr{K}$ and $p \in \{0,1\}^\ell$. *Choose* an arbitrary **seed** $s \in X_e$ and compute the sequence

   $$r = b(s)b(f_e(s)) \dots b(f_e^{\ell-1}(s))$$

   together with $f_e^\ell(s) \in X_e$. Define

   $$\mathscr{E}_e(p) = f_e^\ell(s) \cdot (p + r),$$

   where, as customary, $+$ is the bitwise addition and $\cdot$ the concatenation[4] of bits.
   Let now $d = (e, t_e) \in \mathscr{K}'$ and $c = s' \cdot c'$ with $c' \in \{0,1\}^\ell$. Use $s' = f_e^\ell(s)$ and the trapdoor information t_e to recursively compute $f_e^{\ell-1}(s), \dots, f_e(s), s$. Now compute

   $$r = b(s)b(f_e(s)) \dots b(f_e^{\ell-1}(s))$$

   and return $\mathscr{D}_d(c) = c' + r$.

Definition 7.30. The Blum-Goldwasser construction applied to the Rabin function is called the **Blum-Goldwasser cryptosystem**. □

Theorem 7.31. The Blum-Goldwasser cryptosystem is an asymmetric probabilistic cryptosystem where

1. The FACTORING assumption implies **ASYMMETRY**[5] (i.e., the secret key cannot be computed in polynomial time using the public key).
2. The QR assumption implies IND-CPA.[6]

□

Proof.

1. By definition, computing $d = (n, t_n)$ means factoring the Blum number n.
2. Let $p_1, p_2 \in \{0,1\}^\ell$ and $c_i = \mathscr{E}_e(p_i)$. The QR assumption, Theorem 7.26 (2), and Theorem 5.73 imply that the construction of r defines a CSPRBG. Hence an attacker cannot distinguish between $p_1 + r_1$ and $p_2 + r_2$ (even if $f_e^\ell(s)$ is known; not proven here). ■

[3] Recall that for an asymmetric cryptosystem to satisfy IND it must be probabilistic with $\mathscr{E}$ multi-valued.
[4] $f^\ell(s)$ stands for its bit-coding.
[5] The negation of ASYMMETRY is called "total break". This property only makes sense for public key cryptosystems.
[6] cf. Definitions 2.24 and 2.29 and Remark 2.28

Chapter 8

Public Key Cryptosystems

Now we are ready to investigate asymmetric, i.e. public key cryptosystems in more detail. Recall what this means: the private key cannot be computed in polynomial time using the public key.

8.1 RSA

RSA was one of the first practical public key cryptosystems and is nowadays widely used. Its creators Ron Rivest, Adi Shamir and Leonard Adleman publicly described RSA in 1977.

The main idea of RSA boils down to the following statement:

Lemma 8.1. Let $n, e \in \mathbb{N}$ with $\gcd(e, \varphi(n)) = 1$. Then

$$f_e : (\mathbb{Z}/n\mathbb{Z})^* \to (\mathbb{Z}/n\mathbb{Z})^*, a \mapsto a^e$$

is a permutation with inverse f_d, where $de \equiv 1 \bmod \varphi(n)$. □

Proof. By definition $\varphi(n) = |(\mathbb{Z}/n\mathbb{Z})^*|$. The extended Euclidian division algorithm yields the BÉZOUT identity $de+\lambda\varphi(n) = 1$. Since $a^{\varphi(n)} = 1$ for $a \in (\mathbb{Z}/n\mathbb{Z})^*$ by LAGRANGE's theorem we conclude that

$$a = a^1 = a^{de+\lambda\varphi(n)} = (a^e)^d.$$

■

Recall that for $n = pq$ with p and q distinct primes it follows that $\varphi(n) = (p-1)(q-1)$.

Example 8.2 (RSA function)**.** Define the set

$$I := \{(n = pq, e) \mid p, q \text{ are distinct primes and } \gcd(e,(p-1)(q-1)) = 1\}.$$

For $i = (n = pq, e) \in I$ we define

1. $f_i : a \mapsto a^e \bmod n$ (RSA function resp. encryption).
2. $de \equiv 1 \bmod (p-1)(q-1)$.
3. $g_i : y \mapsto y^d \bmod n$ (inverse function resp. decryption).

□

Definition 8.3. The **RSA problem (RSAP)** is the problem of inverting the RSA function. The **RSA assumption** is that RSAP $\notin$ BPP. □

Remark 8.4.

1. The RSAP reduces to FACTORING, i.e., the RSA assumption is stronger than the FACTORING assumption.
2. Under the RSA assumption: The RSA function is a OWP with trapdoor and hardcore bit $b =$ par (parity bit; without proof). The Blum-Goldwasser construction yields, as for the Rabin function, a probabilistic asymmetric cryptosystem satisfying IND-CPA .

□

Definition 8.5. The **RSA cryptosystem** is defined as follows: For a given $n \in \mathbb{N}_{>0}$ we define

1. $\mathscr{P}_n = \{0,1\}^k$, $\mathscr{C} = \{0,1\}^{k+1}$ where $k = \lfloor \lg n \rfloor$.
2. $\mathscr{K}_n = I = \{(n = pq, e) \mid p, q$ are distinct primes and $\gcd(e, (p-1)(q-1)) = 1\}$ as above.
3. $\mathscr{K}'_n = \{(n = pq, d, e) \mid p, q$ distinct primes, $|p|, |q| \approx \frac{k}{2},\ |pq| \geq k,\ de \equiv 1 \bmod \varphi(n)\}$.
4. $\kappa_n : (n, d, e) \mapsto (n, e) \in \mathscr{K}$.
5. $\mathscr{E}_{(n,e)}(x) = x^e \bmod n$.
6. $\mathscr{D}_{(n,d,e)}(y) = y^d \bmod n$.

Note that in practice one defines a security level k and then defines based on this p, q, and n. □

Remark 8.6. We now list the security properties of the RSA cryptosystem (assuming a CPA attack, which is natural for public cryptosystems):

1. p, q and $\varphi(n)$ must remain secret.
2. RSA assumption $\implies$ OW $\implies$ ASYMMETRY.
3. IND is not satisfied since the cryptosystem is deterministic.
4. NM is not satisfied since the cryptosystem is *multiplicative*: $(ab)^e = a^e b^e$ (see below).

□

Example 8.7. Let $p = 11$, $q = 23$, and $e = 3$. Then $n = pq = 253$, $k = \lfloor \lg n \rfloor = 7$, $\varphi(n) = (p-1)(q-1) = 10 \cdot 22 = 220$, and $d = 147$ with $ed = 441 \equiv 1 \bmod 220$. For $m = 0110100 = (52)_{10}$ we compute

$$c = \mathscr{E}_{(n,e)}(m) = 52^3 = 193 \bmod 253 = (1100001)_2.$$

Now we decrypt via

$$\mathscr{D}_{(253,147,3)}(c) = 193^{147} \equiv 52 \bmod 253 = m.$$

Violating the NM: To shift m one position to the left we manipulate c to $c' = \mathscr{E}_{(253,3)}(2) \cdot c = 2^3 \cdot 193 = 26 \bmod 253$. Then $\mathscr{D}_{(253,147,3)}(c) = 26^{147} \equiv 104 \bmod 253 = (1101000)_2$. □

In analogy with the trivial statement of Theorem 7.31.(1) for the Blum-Goldwasser cryptosystem we prove:

Theorem 8.8. The FACTORING assumption implies the ASYMMETRY of the RSA cryptosystem, i.e., the secret key d cannot be computed in polynomial time using the public key (n, e). □

Proof. Assume that we can compute the secret key d in polynomial time. We need to show that we can then also factor n using the knowledge of $(n, d, e) \in \mathscr{K}'$:
The Chinese Remainder Theorem provides an *isomorphism*

$$(\mathbb{Z}/n\mathbb{Z})^* \to (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^*, \ a \bmod n \mapsto (a \bmod p, a \bmod q).$$

In particular $\operatorname{ord}_{(\mathbb{Z}/n\mathbb{Z})^*}(a) = \operatorname{lcm}(\operatorname{ord}_{(\mathbb{Z}/p\mathbb{Z})^*}(a), \operatorname{ord}_{(\mathbb{Z}/q\mathbb{Z})^*}(a))$. The idea is to use the following trivial equivalence

$$c \equiv 1 \bmod p, \ c \not\equiv 1 \bmod q \iff n > \gcd(c-1, n) = p > 1.$$

to factor n. So our goal is to construct such an element c.
For any a we have that

$$\begin{array}{lcl} \operatorname{ord}_{(\mathbb{Z}/p\mathbb{Z})^*}(a) & \mid & p-1, \\ \operatorname{ord}_{(\mathbb{Z}/q\mathbb{Z})^*}(a) & \mid & q-1, \text{ and} \\ \operatorname{ord}_{(\mathbb{Z}/n\mathbb{Z})^*}(a) & \mid & (p-1)(q-1) = \varphi(n) \mid ed-1. \end{array}$$

Write $ed - 1 = 2^s t$ with t odd. Then $(a^t)^{2^s} = 1$, hence $\operatorname{ord}_{(\mathbb{Z}/n\mathbb{Z})^*}(a^t) \mid 2^s$. Choose randomly an element $a \in (\mathbb{Z}/n\mathbb{Z})^*$ and set $b := a^t$. Then

$$\operatorname{ord}_{(\mathbb{Z}/p\mathbb{Z})^*}(b) = 2^i \text{ and } \operatorname{ord}_{(\mathbb{Z}/q\mathbb{Z})^*}(b) = 2^j \text{ with } i, j \leq s.$$

If $i \neq j$, or w.l.o.g. $i < j$, then $c := b^{2^i} \equiv 1 \bmod p$ and $c \not\equiv 1 \bmod q$ and we get the factorization

$$p = \gcd(c-1, n).$$

We now prove that $i \neq j$ for (at least) half of all $a \in (\mathbb{Z}/n\mathbb{Z})^*$ (recall, $b := a^t$). The choice of a primitive element $g \in (\mathbb{Z}/p\mathbb{Z})^*$ yields an isomorphism $(\mathbb{Z}/(p-1)\mathbb{Z}, +) \to (\mathbb{Z}/p\mathbb{Z})^*$, $x \mapsto g^x$. As above, $\operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(1) = p - 1 \mid 2^s t$ and $\operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(t) \mid 2^s$. Using the identification

$$(\mathbb{Z}/n\mathbb{Z})^* \cong (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^* \cong (\mathbb{Z}/(p-1)\mathbb{Z}, +) \times (\mathbb{Z}/(q-1)\mathbb{Z}, +)$$

it is thus equivalent to show that the inequality $\operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(xt) \neq \operatorname{ord}_{(\mathbb{Z}/(q-1)\mathbb{Z},+)}(yt)$ holds for (at least) half of all pairs $(x, y) \in (\mathbb{Z}/(p-1)\mathbb{Z}, +) \times (\mathbb{Z}/(q-1)\mathbb{Z}, +)$: Let $\operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(t) = 2^k$ and $\operatorname{ord}_{(\mathbb{Z}/(q-1)\mathbb{Z},+)}(t) = 2^\ell$. Note that

$$\begin{array}{lcll} \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(t) & = & \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(xt) & \text{for all } x \text{ odd (trivial).} \\ \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(t) & > & \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(xt) & \text{for all } x \text{ even (trivial).} \end{array}$$

Clearly, the same holds for y and $q - 1$.

Now we distinguish two cases:

$k \neq \ell$**:** Again, w.l.o.g. let $\ell < k$. Then for all (x, y) with x odd we obtain:

$$\operatorname{ord}_{(\mathbb{Z}/(q-1)\mathbb{Z},+)}(yt) \leq \operatorname{ord}_{(\mathbb{Z}/(q-1)\mathbb{Z},+)}(t) = 2^\ell < 2^k = \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(t) = \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(xt).$$

So this inequality holds for at least half of the pairs (x, y), namely those where x is odd.

$k = \ell$: If x is odd and y is even, then

$$\operatorname{ord}_{(\mathbb{Z}/(q-1)\mathbb{Z},+)}(yt) < \operatorname{ord}_{(\mathbb{Z}/(q-1)\mathbb{Z},+)}(t) = 2^k = \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(t) = \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(xt).$$

If x is even and y is odd, then

$$\operatorname{ord}_{(\mathbb{Z}/(q-1)\mathbb{Z},+)}(yt) = \operatorname{ord}_{(\mathbb{Z}/(q-1)\mathbb{Z},+)}(t) = 2^k = \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(t) > \operatorname{ord}_{(\mathbb{Z}/(p-1)\mathbb{Z},+)}(xt).$$

So this inequality holds for at least half of the pairs (x, y), namely those where $x \not\equiv y \bmod 2$.

■

Example 8.9 (Example 8.7 continued)**.** As above let $n = 253 = 11 \cdot 23$, $e = 3$, $d = 147$, $ed - 1 = 220 = 2^2 \cdot 55$. So $s = 2$ and $t = 55$. Try $a = 2$: $b = a^t = 2^{55} \equiv 208 \bmod 253$. Compute $\gcd(b^{2^i} - 1, n)$ for $i = 0, 1 < s = 2$: $\gcd(208 - 1, 253) = 23$. □

Summing up, we get the following implications of security *assumptions*[1] for the RSA cryptosystem (under a CPA attack):

$$\begin{tikzcd}
\boxed{\text{OW}} \arrow[r] & \boxed{\text{ASYMMETRY}} \\
\boxed{\text{RSA}} \arrow[u] \arrow[r] & \boxed{\text{FACTORING}} \arrow[u]
\end{tikzcd}$$

Remark 8.10. A note on the key lengths: In 2009 a number of 768 bits was factored ([KAF$^+$10]). A 1024 bit number is assumed to be factored until 2020. It is nowadays believed that a key length of 2048 bit in RSA is secure for a longer time. Also have a look at `https://www.keylength.com/` where the key length advices of different institutes (e.g. NSA, BSI) can be compared.

To give you a bit of a feeling for the smaller numbers, here follow the timings for factoring numbers on my laptop (Intel Core i7-5557U @ 3.10 GHz):

Bits	Time
128	< 2 sec
192	< 15 sec
256	< 30 min

□

[1] ... not the problems. For the “hardness of the problems” you have to invert the arrows.

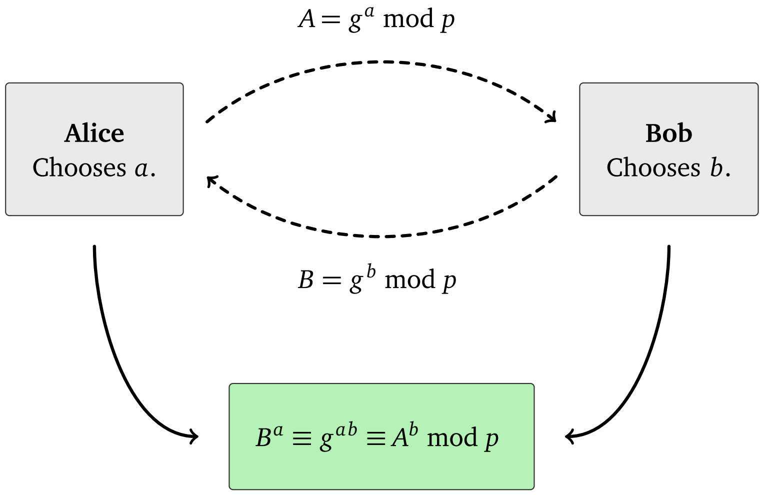


Figure 8.1: Diffie-Hellman Key Exchange

8.2 ElGamal

Recall: Let p be a prime and g be a generator of $(\mathbb{Z}/p\mathbb{Z})^*$. The problem of inverting $\exp_g : (\mathbb{Z}/p\mathbb{Z})^* \to (\mathbb{Z}/p\mathbb{Z})^*$, $x \mapsto g^x$ is the discrete logarithm problem (DLP). $\exp_g$ is a OWP[2] under the DL assumption. We don't have candidates for a trapdoor.

Let us illustrate again, how we introduced the Diffie-Hellman key exchange in Example 1.1 (3) in Figure 8.1:

Definition 8.11. Let p be a prime number and $(\mathbb{Z}/p\mathbb{Z})^* = \langle g \rangle$.

1. The problem of computing g^{ab} given g^a and g^b is called **Diffie-Hellman problem (DHP)**
2. The **Diffie-Hellman or DH assumption** is that DHP $\notin$ BPP.

□

Remark 8.12. The DHP reduces to the DLP, i.e., the DH *assumption* is stronger than the DL assumption. The equivalence is unknown. □

Definition 8.13. The **ElGamal cryptosystem** is defined by

1. $\mathscr{P} = \{0,1\}^k$ and $\mathscr{C} = \{0,1\}^{2(k+1)}$.
2. $\mathscr{K}' = \{(p, g, a) \mid p \text{ prime}, \ \langle g \rangle = (\mathbb{Z}/p\mathbb{Z})^*, \ 2^k < p < 2^{k+1}, \ a \in \{0, \ldots, p-2\}\}$.
3. $\kappa : (p, g, a) \mapsto (p, g, g^a) \in \mathscr{K}$.

We encode $\{0,1\}^k \subset (\mathbb{Z}/p\mathbb{Z})^* \subset \{0,1\}^{k+1}$. So we "replace" $\mathscr{P}$ by $(\mathbb{Z}/p\mathbb{Z})^*$ and $\mathscr{C}$ by $(\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/p\mathbb{Z})^*$. For $e = (p, g, A = g^a)$ and $d = (p, g, a)$ we define

1. $\mathscr{E}_e(x) := (g^b, A^b x)$, where $b \in \{0, \ldots, p-2\}$ is chosen randomly.

[2] $g^0 = g^{p-1} = 1$.

2. $\mathscr{D}_d(y,z) := y^{-a}z$.

Of course, a has to be kept secret. □

Proof of correctness. $y^{-a}z = (g^b)^{-a}A^b x = g^{-ba+ab}x = x$. ■

Example 8.14. Take $p = 23$ and $g = 7$. For $a = 6$, i.e., $d = (23,7,6)$ compute $A = 7^6 \equiv 4 \bmod 23$. $e = \kappa(d) = (23,7,4)$. For $x = 7 \in (\mathbb{Z}/23\mathbb{Z})^* = \mathscr{P}$ compute $\mathscr{E}_e(x)$ for different b's:

1. $b = 3$: $\mathscr{E}_e(x) = (7^3, 4^3 \cdot 7) = (21,11) \in (\mathbb{Z}/23\mathbb{Z})^* \times (\mathbb{Z}/23\mathbb{Z})^* = \mathscr{C}$.
2. $b = 2$: $\mathscr{E}_e(x) = (7^2, 4^2 \cdot 7) = (3,20) \in (\mathbb{Z}/23\mathbb{Z})^* \times (\mathbb{Z}/23\mathbb{Z})^* = \mathscr{C}$.

Now verify $\mathscr{D}_d(21,11) = 21^{-6} \cdot 11 \equiv \boxed{7} \bmod 23 \equiv 3^{-6} \cdot 20 = \mathscr{D}_d(3,20)$. □

Remark 8.15. The ElGamal cryptosystem is a probabilistic public key cryptosystem with multi-valued $\mathscr{E}$. Note the following:

1. It satisfies the IND-CPA security model under the DL assumption (without proof).
2. It does not satisfy NM because of its multiplicativity (like RSA, cf. Remark 8.6).

□

Theorem 8.16. Under a CPA attack the (probabilistic public key) ElGamal cryptosystem satisfies

1. OW under the DH assumption.
2. ASYMMETRY under the DL assumption.

□

Proof.

1. Assume we can decrypt ciphertexts, i.e., from the public key information g^a and the ciphertext $(g^b, g^{ab}x)$ we can compute x. Then we can in particular decrypt $(g^b, 1)$ to obtain g^{-ab} and hence g^{ab}. This contradicts the DH assumption.
2. If a CPA adversary (i.e., who has full access to $A = g^a$ and $\mathscr{E}_e$) can compute the secret key information a, then she has already solved the DLP. ■

8.3 The Rabin cryptosystem

Next we recall the Rabin function from Definition 7.27 and Example 7.29. We look into the non-unique decryption process in more detail.

Definition 8.17. The **Rabin cryptosystem** is defined as follows:

1. $\mathscr{P} = \{0,1\}^k$ and $\mathscr{C} = \{0,1\}^{k+1}$.
2. $\mathscr{K}' = \{(p,q) \mid p,q \text{ distinct primes, } p,q \equiv 3 \bmod 4,\ 2^k < pq < 2^{k+1}\}$.[3]
3. $\kappa : (p,q) \mapsto pq \in \mathscr{K} = \{n \in \mathbb{N} \mid 2^k < n < 2^{k+1} \text{ a Blum number}\}$.

[3] Note that one can show that $\mathscr{K}'$ is not empty for "reasonable" choices of k.

We encode $\{0,1\}^k \subset \mathbb{Z}/n\mathbb{Z} \subset \{0,1\}^{k+1}$. So we "replace" $\mathscr{P}$ and $\mathscr{C}$ by $\mathbb{Z}/n\mathbb{Z}$. For $e = n$ and $d = (p,q)$ we define

1. $\mathscr{E}_e(x) := x^2 \bmod n$ (not injective!)
2. $\mathscr{D}_d(y) :=$ the four square roots of $x^2 \bmod n$ (not uniquely determined!) using the Chinese Remainder Theorem and the simple case "$p \equiv 3 \bmod 4$" in the Tonelli-Shanks algorithm from the proof of Theorem 7.22.

□

Note that the Rabin cryptosystem is *not* the Blum-Goldwasser cryptosystem from Definition 7.30.

Example 8.18. Take $p = 3$ and $q = 7$. Then $n = 21$ is the public key of the Rabin cryptosystem. To encrypt the plain text $m = 10 \in \mathbb{Z}/21\mathbb{Z}$ we compute

$$c = m^2 = 10^2 \equiv 16 \bmod 21.$$

To decrypt $c = 16$ using the secret prime factors $p = 3$ and $q = 7$ we compute the four square roots of 16 modulo 21 using the Tonelli-Shanks algorithm from Theorem 7.22: We have

$$16^{\frac{p+1}{4}} = 16 \equiv \boxed{1} \bmod 3 \quad \text{and} \quad 16^{\frac{q+1}{4}} = 16^2 \equiv \boxed{4} \bmod 7.$$

Hence

1. 1 and $-1 \equiv 2 \bmod 3$ are the square roots of 16 modulo 3.
2. 4 and $-4 \equiv 3 \bmod 7$ are the square roots of 16 modulo 7.

With the Chinese Remainder Theorem we get the four combinations

$$(1,4), \quad (1,3), \quad (2,4), \quad (2,3)$$

and finally the four square roots

$$4, \quad 10, \quad 11, \quad 17,$$

among which we search for the (hopefully unique) human readable "plain text" 10. □

Theorem 8.19. For the Rabin cryptosystem the following implications of assumptions hold (under a CPA attack)

$$\mathsf{FACTORING} \implies \mathsf{OW} \implies \mathsf{ASYMMETRY}.$$

□

Proof. If we can decrypt ciphertexts we can choose random elements $x \in \mathbb{Z}/n\mathbb{Z}$ and compute square roots of x^2. By Theorem 7.25 we then obtain a factorization of n, so a total break of the cryptosystem. This proves the first implication. The second implication is trivial. ■

Remark 8.20. One "problem" of the Rabin cryptosystem is the decryption which is in general not unique and a contextual search needs to be done. Note that restricting $\mathscr{P}$ to Q_n does not eliminate the non-uniqueness issue: It is then not clear how to find an injective encoding $\{0,1\}^k \to Q_n$? □

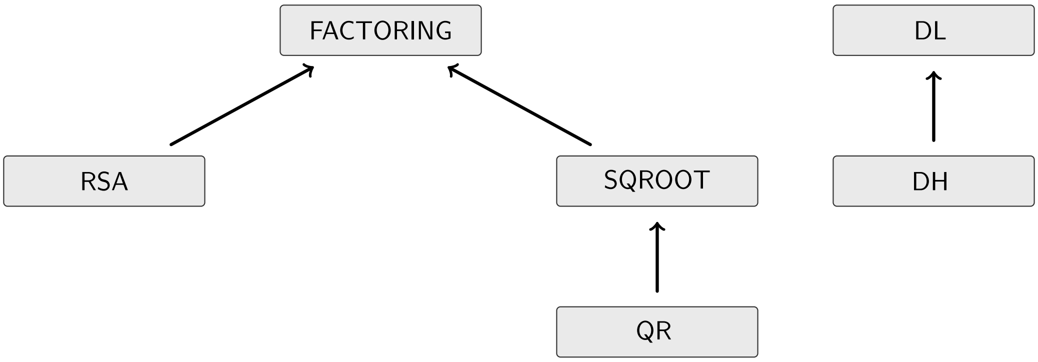
8.4 Security models

Remark 8.21. We know that the following security models are not fulfilled:

1. The Blum-Goldwasser cryptosystem does *not* satisfy the security model IND-CCA: We first recall Definition 7.30: We have an encryption function $\mathcal{E}_e(p) = f_e^\ell(s) \cdot (p + r) = s' \cdot c' =: c$ for $r = b(s)b(f_e(s)) \dots b(f_e^{\ell-1}(s))$ for a random seed s. Now we can adaptively choose another cipher text $\tilde{c}$ by flipping one bit in the "$(p + r)$"-part of c. The decryption oracle gives us now the decrypted plain text $\tilde{p} = \tilde{c}' + r$. Thus we can recover p via applying
$$\tilde{c}' + \tilde{p} + c' = r + c' = p.$$
This attack works since s' is completely independent of c'.
2. The same reasoning as in the proof of Theorem 8.19 shows that the Rabin cryptosystem does *not* fulfill the security model ASYMMETRY-CCA.

□

We have the following hierarchy of *assumptions*



Remark 8.22. It is conjectured that all backward implications in the above diagram hold.[4] These conjectures imply that the RSA, ElGamal, and Blum-Goldwasser cryptosystems do *not* fulfill the security model ASYMMETRY-CCA:

1. RSA =FACTORING $\implies$ RSA $\notin$ ASYMMETRY-CCA.
2. DH =DL $\implies$ ElGamal $\notin$ ASYMMETRY-CCA.
3. QR=SQROOT $\implies$ Blum-Goldwasser $\notin$ ASYMMETRY-CCA (exercise).

□

IND-CCA

One can modify the Blum-Goldwasser cryptosystem in such a way that IND-CCA is fulfilled: For this we need the concept of a **one-way hash function**, which is, roughly speaking, a one-way function $H : \{0,1\}^\bullet \to \{0,1\}^k$ for some $k \in \mathbb{N}$.

[4] Recall that we already know that for Blum numbers the FACTORING assumption $\implies$ SQROOT assumption (cf. Theorem 7.25) holds.

Remark 8.23. The modified Blum-Goldwasser cryptosystem (cf. Definition 7.30)

$$\mathscr{E}_e(p) = f_e^\ell(s + H(y)) \cdot \underbrace{(p + r)}_{y}$$

now satisfies, under the QR assumption, the security model IND-CCA. Thus we cannot apply an attack as done in Remark 8.21 any longer: $s' = f_e^\ell(s + H(y))$ now depends on $c' = p + r = y$. □

Optimal Asymmetric Encryption Padding (OAEP)

We now describe the so-called **optimal asymmetric encryption padding**[5] **(OAEP)** [Wik16d] which is often used to improve the security of public key cryptosystems by preprocessing plaintexts prior to the asymmetric encryption:

1. Fix a security parameter $k \in \mathbb{N}$.
2. Fix $k_0, k_1 \in \mathbb{N}$ with $n := k - k_0 - k_1 > 0$.
3. Fix a CSPRBG
$$G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}.$$
4. Fix a one-way hash function
$$H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0},$$
called the **compression function**.

For an n-bit plaintext p and seed $s \in \{0,1\}^{k_0}$ return the k-bit concatenated string

$$p' = \underbrace{\left((p \cdot \underbrace{0 \ldots 0}_{k_1}) + G(s) \right)}_{=:y \in \{0,1\}^{n+k_1}} \cdot (s + H(y)).$$

Now one can apply the OWP $f : \{0,1\}^k \to \{0,1\}^k$ of public key cryptosystem to the padded message p'.

Definition 8.24. The *probabilistic* public key cryptosystem obtained by applying the RSA function to an OAEP-preprocessed message p' is called the **RSA-OAEP cryptosystem**. □

Assuming the existence of a so-called ideal compression function H one can prove that

Theorem 8.25. The RSA-OAEP cryptosystem satisfies the security model IND-CCA under the RSA assumption. □

[5] German: Polsterung

Chapter 9

Primality tests

In the last chapter we have seen that many public key cryptosystems are based on the fact of finding big prime factors. In this chapter we want to study various sorts of probabilistic and deterministic primality test. Let us denote by $\mathbb{P} \subset \mathbb{N}$ the set of prime numbers.

9.1 Probabilistic primality tests

Fermat test

Recall **Fermat's little theorem**:[1]

Theorem 9.1. For $p \in \mathbb{P}$ it holds that $a^{p-1} \equiv 1 \bmod p$ for all $a \in \mathbb{Z} \setminus p\mathbb{Z}$. □

This yields the so-called **Fermat test**, an elementary probabilistic test for primality, which lies in $\mathcal{O}(\log \log \log n)$:

Let $n \in \mathbb{N}$. If there exists $a \in (\mathbb{Z}/n\mathbb{Z})^*$ with $a^{n-1} \not\equiv 1 \bmod n$ then n is not a prime.

Note that finding an $a \in N_n := ((\mathbb{Z}/n\mathbb{Z}) \setminus \{0\}) \setminus (\mathbb{Z}/n\mathbb{Z})^*$ is hopeless if n is the product of huge primes (compare $n = |\mathbb{Z}/n\mathbb{Z}|$ and $n - \varphi(n) = |N_n| + 1$).

Example 9.2. Let $n = 341$. For

1. $a = 2$: $2^{340} \equiv 1 \bmod 341$.
2. $a = 3$: $3^{340} \equiv 56 \bmod 341$.

Hence, 341 is a composite number and 3 is a witness.[2] □

Definition 9.3. Let $n \in \mathbb{N}$ and $a \in (\mathbb{Z}/n\mathbb{Z})^*$.

1. n is called a **pseudoprime with Fermat nonwitness** a if $a^{n-1} \equiv 1 \bmod n$, i.e., the Fermat test of the primality of n passes for a.
2. If n is a pseudoprime with Fermat nonwitness a but not prime then a is called a **Fermat liar**.
3. If the Fermat test of the primality of n fails for a, i.e., if $a^{n-1} \not\equiv 1 \bmod n$, then a is called a **Fermat witness (for the compositeness) of** n.

[1] This is a special case of Euler's Theorem $a^{\varphi(n)} \equiv 1 \bmod n$, for the case $n = p \in \mathbb{P}$.

[2] German: Zeuge.

4. n is called a **Carmichael number** if n is a composite number without a Fermat witness, i.e., if all $a \in (\mathbb{Z}/n\mathbb{Z})^*$ are Fermat liars.

□

Of course, each prime is a pseudoprime for all $a \in (\mathbb{Z}/n\mathbb{Z})^*$. In 1994 it was proven in [AGP94] that the set of Carmichael numbers is infinite:

$$561 = 3 \cdot 11 \cdot 17, \quad 1105 = 5 \cdot 13 \cdot 17, \quad 1729 = 7 \cdot 13 \cdot 19, \quad \dots$$

Lemma 9.4. Let n be a Carmichael number and $p \in \mathbb{P}$. Then the following statements hold:

1. n is odd.
2. n is square free.
3. $p \mid n \implies p-1 \mid n-1$.
4. n has at least 3 prime factors.

□

Proof.

1. n even $\implies n-1$ odd $\underset{\text{Carmichael}}{\overset{n}{\implies}} -1 = (-1)^{n-1} = 1 \in (\mathbb{Z}/n\mathbb{Z})^* \implies n = 2$ prime $\lightning$ (since 2 as a prime is not Carmichael).

2. Write $n = p^e \cdot n'$, where e is the maximal p-power. Then
$$\varphi(n) = \varphi(p^e)\varphi(n') = p^{e-1}(p-1)\varphi(n').$$
Assume that $p^2 \mid n$. Then it follows:
   a) $p \mid \varphi(n) = |(\mathbb{Z}/n\mathbb{Z})^*| \implies \exists a \in (\mathbb{Z}/n\mathbb{Z})^*$ with $\operatorname{ord}_{(\mathbb{Z}/n\mathbb{Z})^*}(a) = p$.
   b) $p \mid n \implies p \nmid n-1 \overset{\operatorname{ord}(a)=p}{\implies} a^{n-1} \not\equiv 1 \bmod n \implies a$ is a Fermat witness for $n \implies n$ is not a Carmichael number.

3. Let p be a prime divisor of n. Since $a^{n-1} \equiv 1 \bmod n$ it follows that $a^{n-1} \equiv 1 \bmod p$, for all $a \in \mathbb{Z}$ with $\gcd(a, n) = 1$. Since $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic, we can choose a to be a primitive element. Thus $\operatorname{ord} a = |(\mathbb{Z}/p\mathbb{Z})^*|$. Thus we deduce that $p-1 = |(\mathbb{Z}/p\mathbb{Z})^*| \mid n-1$.

4. Clearly, n must have more than one prime factor. Suppose n has two prime factors, $n = pq$, $p, q \in \mathbb{P}$. W.l.o.g. $p > q$ since n is square free. Then $p-1 > q-1$, so $p-1$ does not divide $q-1$. Now
$$(n-1) - (q-1) = n - q = pq - q = (p-1)q.$$
Since $p-1$ does not divide $q-1$ it also cannot divide $n-1$. Since p divides n this contradicts (3) thus n is not a Carmichael number. ■

The existence of infinitely many Carmichael numbers means that we cannot trust the Fermat primality test (unless of course it produces a Fermat witness).

Miller-Rabin test

The Miller-Rabin test makes use of the fact that the equation $a^2 = 1$ has exactly two solutions $a = \pm 1$ over $\mathbb{Z}/n\mathbb{Z}$ if n is a prime (since then $\mathbb{Z}/n\mathbb{Z}$ is a field). The general idea is to improve the Fermat test by successively dividing the power of a chosen element a by 2 and test again, starting with a^{n-1}.

Lemma 9.5 (Miller-Rabin). Let $p \in \mathbb{P}$ and let $a \in (\mathbb{Z}/p\mathbb{Z})^*$. Write $p-1 = 2^s t$ with t odd ($s \geq 0$). Then

$$\begin{aligned} a^t &\equiv \pm 1 \bmod p && \underline{\text{or}} \\ a^{2^r t} &\equiv -1 \bmod p && \text{for } \underline{\text{some}}\ 0 < r < s. \end{aligned}$$

□

Proof. Let $0 \leq s_0 \leq s$ minimal with $a^{2^{s_0} t} \equiv 1 \bmod p$ (recall that $a^{p-1} \equiv 1 \bmod p$). We distinguish two cases:

1. If $s_0 = 0$ then $a^t \equiv 1 \bmod p$.
2. If $s_0 > 0$ then $a^{2^r t} \equiv -1 \bmod p$ with $r = s_0 - 1 \in \{0, \ldots, s-1\}$. ■

Definition 9.6. Let n be a composite number. Write $n - 1 = 2^s t$ with t odd ($s \geq 0$). $a \in (\mathbb{Z}/n\mathbb{Z})^*$ is called a **Miller-Rabin nonwitness** if

$$\begin{aligned} a^t &\equiv \pm 1 \bmod n && \underline{\text{or}} && (\pm 1) \\ a^{2^r t} &\equiv -1 \bmod n && \text{for } \underline{\text{some}}\ 0 < r < s, && (-1) \end{aligned}$$

otherwise a is called a **Miller-Rabin witness (for the compositeness of n).** □

Example 9.7. Consider $n = 561$: $n-1 = 560 = 2^4 \cdot 35$, so $s = 4$ and $t = 35$. For $a = 2$ we compute

$$\left.\begin{array}{rcr} 2^{35} & \equiv & 263 \bmod 561 \end{array}\right\} \not\equiv 1, -1 \bmod 561$$

$$\left.\begin{array}{rcr} 2^{2\cdot 35} & \equiv & 166 \bmod 561 \\ 2^{4\cdot 35} & \equiv & 67 \bmod 561 \\ 2^{8\cdot 35} & \equiv & 1 \bmod 561 \end{array}\right\} \not\equiv -1 \bmod 561$$

So $a = 2$ is a Miller-Rabin witness for Carmichael number 561. □

Remark 9.8. If the **generalized Riemann hypothesis** holds, then n is a prime if one of the conditions (± 1) or (-1) is fulfilled for each $1 < a < 2 \log^2 n$. This turns the probabilistic Miller-Rabin test into a *deterministic* one. See Remark 9.13 below. □

Definition 9.9. For a fixed $n \in \mathbb{N}$ define

$$N := \{\text{Miller-Rabin nonwitness for } n\} \subset (\mathbb{Z}/n\mathbb{Z})^*.$$

□

The idea is to find a subgroup $U \le (\mathbb{Z}/n\mathbb{Z})^*$ with $N \subset U$ and to bound the index $(\mathbb{Z}/n\mathbb{Z})^* : U$ from below away from 1. A natural candidate would be

$$U_0 := \{a \in (\mathbb{Z}/n\mathbb{Z})^* \mid a^{n-1} \equiv 1 \bmod n\} = \{\text{Fermat nonwitness}\} = \ker(x \mapsto x^{n-1}) \le (\mathbb{Z}/n\mathbb{Z})^*.$$

But we know that the index $(\mathbb{Z}/n\mathbb{Z})^* : U_0$ might be 1:

$$U_0 = (\mathbb{Z}/n\mathbb{Z})^* \iff n \text{ is a prime or a Carmichael number.}$$

Lemma 9.10. Let $n = p^\alpha$ for $\alpha \ge 2$. Then $(\mathbb{Z}/n\mathbb{Z})^* : U_0 \ge p$. □

Proof. $p \mid p^{\alpha-1}(p-1) = \varphi(n) = |(\mathbb{Z}/n\mathbb{Z})^*|$. Then there exists an $a \in (\mathbb{Z}/n\mathbb{Z})^*$ with $\operatorname{ord}(a) = p$. Furthermore, $p \mid n \implies p \nmid n-1 \implies a^{n-1} \not\equiv 1 \bmod n \implies a \notin U_0$. The same holds for $a^2, \ldots, a^{p-1}$. Hence $U_0, aU_0, a^2U_0, \ldots, a^{p-1}U_0 \in (\mathbb{Z}/n\mathbb{Z})^*/U_0$ are pairwise different and $(\mathbb{Z}/n\mathbb{Z})^* : U_0 \ge p$. ■

Theorem 9.11. Let n be a composite odd number with $3 \nmid n$. Then $|N| \le \frac{\varphi(n)}{4} < \frac{n}{4}$. □

Proof. Again write $n - 1 = 2^s t$ with t odd ($s \ge 1$). Set $N_{-1} := \{a \in N \mid a^t \equiv 1 \bmod n\}$ and $N_i := \{a \in N \mid a^{2^i t} \equiv -1 \bmod n\}$ for $0 \le i < s$. Then $N = \bigcup_{i=-1}^{s-1} N_i \neq \emptyset$ (since $-1 \in N_0 \neq \emptyset$). Set $r := \max\{i \mid N_i \neq \emptyset\} \in \{0, \ldots, s-1\}$ and $m := 2^r t$. In particular, $m \mid \frac{n-1}{2} = \frac{2^s t}{2}$ (because $r < s$). For all $a \in N$ it holds:

$$a^m \equiv \begin{cases} -1 & \text{if} \quad a \in N_r, \\ 1 & \text{if} \quad a \in N_i \text{ for } i < r. \end{cases} \tag{9.1}$$

Consider the group endomorphism $f : (\mathbb{Z}/n\mathbb{Z})^* \to (\mathbb{Z}/n\mathbb{Z})^*$, $a \mapsto a^m$. Let $n = p_1^{\alpha_1} \cdots p_u^{\alpha_u}$ be the factorization of n as the product of distinct prime powers. The Chinese Remainder Theorem yields the isomorphism

$$(\mathbb{Z}/n\mathbb{Z})^* \cong (\mathbb{Z}/p_1^{\alpha_1}\mathbb{Z})^* \times \cdots \times (\mathbb{Z}/p_u^{\alpha_u}\mathbb{Z})^*,$$

identifying $a \in (\mathbb{Z}/n\mathbb{Z})^*$ with the u-tuple $(a \bmod p_1^{\alpha_1}, \ldots, a \bmod p_u^{\alpha_u})$. We now use this isomorphism to define the following chain of subgroups

$$\begin{array}{llll}
(\mathbb{Z}/n\mathbb{Z})^* & & & \\
\quad \rotatebox{90}{$\le$} & & & \\
U_0 & := & \ker(x \mapsto x^{n-1}) & = \{a \in (\mathbb{Z}/n\mathbb{Z})^* \mid a^{n-1} \equiv 1 \bmod n\}, \\
\quad \rotatebox{90}{$\le$} & & & \\
U_1 & := & f^{-1}(\{(\pm 1, \ldots, \pm 1)\}) & = \{a \in (\mathbb{Z}/n\mathbb{Z})^* \mid a^m \equiv \pm 1 \bmod p_i^{\alpha_i},\ 1 \le i \le u\} \\
\quad \rotatebox{90}{$\le$} & & & \\
U_2 & := & f^{-1}(\{\pm 1\}) & = \{a \in (\mathbb{Z}/n\mathbb{Z})^* \mid a^m \equiv \pm 1 \bmod n\}, \\
\quad \rotatebox{90}{$\le$} & & & \\
U_3 & := & f^{-1}(\{1\}) & = \{a \in (\mathbb{Z}/n\mathbb{Z})^* \mid a^m \equiv 1 \bmod n\}.
\end{array}$$

U_1 is a subgroup of U_0 since $m \mid \frac{n-1}{2}$ (see above). The remaining inclusions are obvious as the preimages of inclusions of a chain of subgroups. Since $N \subset U_2$ we want to bound the index $(\mathbb{Z}/n\mathbb{Z})^* : U_2$ from below away from 1. We claim that

$$(\mathbb{Z}/n\mathbb{Z})^* : U_2 \ge 4.$$

To see this we first have to prove that $\{(\pm 1, \ldots, \pm 1)\} \le \operatorname{im} f$ as a subgroup:
Choose a $b \in N_r \neq \emptyset$, then $f(b) = b^m \equiv -1 \bmod n$, hence, $b^m \equiv -1 \bmod p_i^{\alpha_i}$ for all $i = 1, \ldots, u$.

Now let y be an arbitrary element of the elementary Abelian subgroup[3] $\{(\pm 1, \dots, \pm 1)\} \cong (\mathbb{F}_2^u, +)$, w.l.o.g. we can assume that $y = (1, \dots, 1, -1, \dots, -1)$. Then $x := (1, \dots, 1, b, \dots, b)$ is a preimage of y under f, i.e., $f(x) = y$. Summing up:

$$U_3 \underbrace{\underbrace{<}_{2} U_2 \underbrace{\leq}_{2^{u-1}}}_{2^u} U_1 \leq U_0 \leq (\mathbb{Z}/n\mathbb{Z})^*.$$

We now distinguish three cases:

$u \geq 3$: $U_2 \underbrace{<}_{\geq 4} U_1 \leq U_0 \leq (\mathbb{Z}/n\mathbb{Z})^*$.

$u = 2$: $U_2 \underbrace{\underbrace{<}_{2} U_1 \leq U_0 \underbrace{<}_{\geq 2}}_{\geq 4} (\mathbb{Z}/n\mathbb{Z})^*$, by Lemma 9.4.(4).

$u = 1$: $U_2 \underbrace{=}_{1} U_1 \leq U_0 \underbrace{<}_{\geq p \geq 5} (\mathbb{Z}/n\mathbb{Z})^*$, by Lemma 9.10 and the assumptions on n.

This finishes the proof. ■

The proof provides a probabilistic primality test in $\mathcal{O}(\log \log \log n)$. If the Miller-Rabin test passes for i randomly chosen different a's then the probability of n being prime is greater than $1 - \left(\frac{1}{4}\right)^i$.

Example 9.12. Now we demonstrate the difference between the Fermat test and the Miller-Rabin test on a trival example. Let $n := 185 = 5 \cdot 37$. Now $n - 1 = 184 = 2^3 \cdot 23 = 8 \cdot 23$, thus $t = 23$ and $s = 3$:

a	1	-1	43		36	6	2
	$\in N_{-1}$	$\in N_0$	$\in N_1 = N_r$	$N_2 = \emptyset$			
a^t	1	-1	$\not\equiv \pm 1$		$\not\equiv \pm 1$	$\not\equiv \pm 1$	$\not\equiv \pm 1$
a^{2t}	1	1	-1		1	$\not\equiv -1$	$\not\equiv -1$
a^{4t}	1	1	1		1	1	$\not\equiv -1$
	Miller-Rabin nonwitnesses				Miller-Rabin witnesses		
a^{8t}	1	1	1		1	1	$\not\equiv 1$
	Fermat nonwitnesses						Fermat witnesses

□

[3] $2 \nmid n \implies 1 \not\equiv -1 \bmod p_i^{\alpha_i}$ for all $i = 1, \dots, u$.

Remark 9.13. One can prove the following nice statements:

$$\begin{array}{lll}
n < 2047 & \text{is prime} \iff & (\pm 1) \text{ or } (-1) \text{ is fulfilled for } a = 2. \\
n < 1373653 & \text{is prime} \iff & (\pm 1) \text{ or } (-1) \text{ is fulfilled } \forall a \in \{2,3\}. \\
\vdots & & \\
n < \underbrace{341550071728321}_{>3.4\cdot 10^{14}} & \text{is prime} \iff & (\pm 1) \text{ or } (-1) \text{ is fulfilled } \forall a \in \{2,\ldots,17\}, \\
& & \text{i.e., for all of the first 7 primes.}
\end{array}$$

For such n's the probabilistic Miller-Rabin test becomes a deterministic one. □

9.2 Deterministic primality tests

The AKS-algorithm

In this subsection we sketch the **AKS**-test, which was proposed by **A**grawal and his master students **K**ayal and **S**axena in 2002. It was published in 2004: [AKS04]. The AKS-test is the first deterministic polynomial runtime primality test.

Lemma 9.14. Let $n \in \mathbb{N} \setminus \{1\}$ and $a \in \mathbb{Z}$ coprime to n. Then[4]

$$n \text{ is prime} \iff (x+a)^n \equiv x^n + a \bmod n.$$

□

Proof.

$\Rightarrow$: Let $n \in \mathbb{P}$. Then $n \mid \binom{n}{i}$ for all $0 < i < n$. Further, $a^n \equiv a \bmod n$ (recall, n is prime). Then $(x+a)^n = \sum_{i=0}^{n} \binom{n}{i} a^i x^{n-i} \equiv x^n + a^n \equiv x^n + a \bmod n$.

$\Leftarrow$: Let $(x+a)^n = \sum_{i=0}^{n} \binom{n}{i} a^i x^{n-i} \equiv x^n + a \bmod n$ (*). Let p be a prime divisor of n such that $p < n$. Then $\binom{n}{p} := \frac{n(n-1)\cdots(n-p+1)}{p(p-1)\cdots 1}$ is not divisible by n, since $p \mid n$ and $p \nmid (n-1),\ldots,(n-p+1)$. Together with $\gcd(a,n) = 1$ this implies that $\binom{n}{p} a^p \not\equiv 0 \bmod n$. Hence $n = p$ by (*). ■

The idea is to consider the equation

$$(x+a)^n \equiv x^n + a \bmod (n, x^r - 1),$$

for a fixed r, i.e., reduce the coefficients modulo n and the polynomial modulo $x^r - 1$. Clearly, by the above lemma we know: If n is a prime number, then $(x+a)^n \equiv x^n + a \bmod (n, x^r - 1)$ always holds. The following criterion now states that also the (slightly relaxed) converse is true. Thus we can use $(x+a)^n \not\equiv x^n + a \bmod (n, x^r - 1)$ for testing the compositeness of numbers.

We state without proof:

Theorem 9.15 (AKS-criterion). Let $2 < n \in \mathbb{N}$ and $r \in \mathbb{N}$ coprime to n. Further let $1 < s \in \mathbb{N}$ with $\gcd(a,n) = 1$ for all $a = 1,\ldots,s$ and

$$\binom{\varphi(r)+s-1}{s} > n^{2d\left\lfloor\sqrt{\frac{\varphi(r)}{d}}\right\rfloor} \quad \text{for all } d \mid \frac{\varphi(r)}{t}, \tag{AKS}$$

[4] The right hand side is an identity of polynomials in x.

where $t := |\langle n\rangle_{(\mathbb{Z}/r\mathbb{Z})^*}|$. If

$$(x+a)^n \equiv x^n + a \bmod (n, x^r - 1), \quad \text{for all } a = 1, \ldots, s,$$

then n is a prime *power*. □

To justify an early step in the AKS algorithm below we need a simple corollary of the following Lemma which we also state without proof:

Lemma 9.16 (Chebyshev[5])**.** For $k \geq 2$

$$\prod_{\substack{p\in\mathbb{P}\\ p\leq 2k}} p > 2^k.$$

□

Corollary 9.17. Let $N \geq 2$ be a natural number of bit length $k := \lceil \lg N \rceil$. Then there exists a prime $p \leq 2k$ with $p \nmid N$. □

Proof. $N < 2^k$, by definition of k. Now use the previous lemma. ■

The following version of the AKS-algorithm is due to Bernstein and Lenstra.

Algorithm 9.18. Let $n \in \mathbb{N} \setminus \{1\}$ be an odd number.

1. Compute (the factors of) $N := 2n(n-1)(n^2-1)\cdots(n^{4\lceil \lg n\rceil^2} - 1)$ of bit length $k := \lceil \lg N \rceil$.
2. Find the smallest prime $r \leq 2k$ with $r \nmid N$. If, before reaching the smallest prime r, you discover that
   a) n is a prime ($n < r$) then **return**: n prime.
   b) a prime $p \mid n$ ($p < r$) then **return**: n composite.
3. If there is an element $a \in \{1, \ldots, r\}$ with $(x+a)^n \not\equiv x^n + a \bmod (n, x^r - 1)$ then **return**: n composite.
4. If there is an element $a \in \{1, \ldots, \log_r n\}$ with $\sqrt[a]{n} \in \mathbb{N}$ then **return**: n composite.
5. **return**: n prime.

□

Theorem 9.19. Algorithm 9.18 is correct and has polynomial runtime, i.e., it lies in $\mathcal{O}(f(\ell))$, where f is a polynomial in $\ell := \lceil \lg n \rceil$. □

Proof.

1. The factors $n-1, n^2-1, \ldots, n^{4\lceil \lg n\rceil^2} - 1$ can be computed with less than $4\ell^2 \lg\left(4\ell^2\right)$ multiplications. Further $\lg N \leq 1 + \lg n + (\lg n)\sum_{i=1}^{4\ell^2} i \leq 1 + \ell + \ell\frac{(4\ell^2+1)4\ell^2}{2}$, in particular, the bit length $k := \lceil \lg N \rceil$ is polynomial in ℓ.

[5] German: Tschebyscheff

2. The runtime of listing all primes $\leq 2k$ is a polynomial in ℓ. If case (a) or (b) occur then the algorithm terminates.

3. Claim (i): $t := |\langle n \rangle_{(\mathbb{Z}/r\mathbb{Z})^*}| > 4\ell^2$. Proof: If not then there would exist an $i \in \{1, \dots, 4\ell^2\}$ with $n^i \equiv 1 \bmod r \implies r \mid n^i - 1 \mid N$ ↯.
   Consider the AKS-criterion for $s = r$. The previous steps guarantee that $\gcd(a, n) = 1$ for all $a = 1, \dots, r = s$ (recall, $r \nmid n$ since $n \mid N$).

   Claim (ii): The (AKS) inequality is fulfilled. Proof: From $d \leq \frac{\varphi(r)}{t} \overset{\text{Claim (i)}}{<} \frac{\varphi(r)}{4\ell^2}$ it follows that

   $$2d \left\lfloor \sqrt{\frac{\varphi(r)}{d}} \right\rfloor \leq 2d \sqrt{\frac{\varphi(r)}{d}} = \sqrt{4d\varphi(r)} \overset{\text{here}}{<} \frac{\varphi(r)}{\ell} \leq \frac{\varphi(r)}{\lg n}. \tag{*}$$

   Further $2 \mid N \implies r \geq 3 \implies \varphi(r) = r - 1 \geq 2 \implies$

   $$\binom{\varphi(r) + s - 1}{s} = \binom{\varphi(r) + r - 1}{r} = \binom{2\varphi(r)}{\varphi(r) + 1} \geq 2^{\varphi(r)} = n^{\frac{\varphi(r)}{\lg n}} \overset{(*)}{>} n^{2d\left\lfloor \sqrt{\frac{\varphi(r)}{d}} \right\rfloor}.$$

   The AKS-criterion (Theorem 9.15) can now be applied proving the correctness of step (3). Note that exponentiating with n is polynomial in ℓ.

4. If this step is reached then n is a prime power by the AKS-criterion. That this step is also polynomial in ℓ is an easy exercise. ■

Chapter 10

Integer Factorization

Recall Remark 8.10: In 2009 the factoring of a number of 768 bits was performed. Still, the following quote from [KAF$^+$10] shows how hard this task was:

> The following effort was involved. We spent half a year on 80 processors on polynomial selection. This was about 3% of the main task, the sieving, which was done on many hundreds of machines and took almost two years. ...

So factorization is still hard, there is until today no polynomial algorithm known.

10.1 Pollard's $p-1$ method

Pollard invented this method in 1974.

Definition 10.1. Let $B \in \mathbb{N}$. An $n \in \mathbb{N}$ is called

1. ***B*-smooth** if all its prime divisors are less than or equal to B.
2. ***B*-powersmooth** if all its prime *power* divisors are less than or equal to B.

□

We now describe **Pollard's $p-1$ method** to factor a composite integer $n \in \mathbb{N}$ where p is a prime divisor of n:

1. If $\gcd(a,n) = 1$ for an $a \in \mathbb{Z}$ then $a^{p-1} \equiv 1 \bmod p$ (Fermat's little theorem).
2. Assume $p-1$ is B-powersmooth for a "small" bound $B \in \mathbb{N}$. Then $p-1 \mid \operatorname{lcm}\{1,\ldots,B\}$ and hence $a^{\operatorname{lcm}\{1,\ldots,B\}} \equiv 1 \bmod p$, or equivalently $p \mid a^{\operatorname{lcm}\{1,\ldots,B\}} - 1$. In particular:
$$\gcd(a^{\operatorname{lcm}\{1,\ldots,B\}} - 1, n) > 1$$
and we have found a divisor[1] of n.
3. A good heuristic value for B is $B \geq n^{\frac{1}{2}\left(1-\frac{1}{e}\right)} \approx n^{0.316}$. So for a fixed B the method should be able to cover all $n \leq B^{2\frac{e}{e-1}} \approx B^{3.164}$. Typically one chooses $B \approx 10^6$ which allows handling numbers $n \leq 10^{19}$.

[1] Of course, the gcd might be n.

Exercise 10.2. Describe a factorization algorithm using the above idea. Use your algorithm to factor 1633797455657959. *Hint:* Try with a very small $B \approx 20$. □

10.2 Pollard's ρ method

Pollard invented this method in 1975, shortly after the $p-1$ method.

Let n be a composite number and $x_0, x_1, \ldots$ be a sequence in $\mathbb{Z}/n\mathbb{Z}$. For a prime divisor p of n set $y_k := x_k \bmod p$. Since $\mathbb{Z}/p\mathbb{Z}$ is finite, two y's, say y_μ and $y_{\mu+\lambda}$ ($\mu, \lambda \in \mathbb{N}, \lambda > 0$), will eventually coincide:

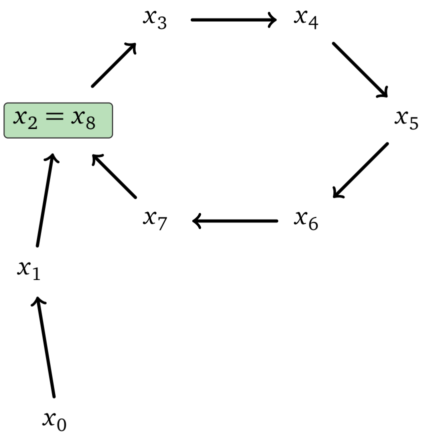
$$y_{\mu+\lambda} = y_\mu \in \mathbb{Z}/p\mathbb{Z}, \text{ or equivalently, } p \mid x_{\mu+\lambda} - x_\mu.$$

But then $d := \gcd(x_{\mu+\lambda} - x_\mu, n) > 1$ is a factor of n. The trivial (improbable) case $d = n$ only occurs if already $x_{\mu+\lambda} = x_\mu \in \mathbb{Z}/n\mathbb{Z}$.

If the sequence $x_0, x_1, \ldots \in \mathbb{Z}/n\mathbb{Z}$ is chosen randomly then $y_0, y_1, \ldots$ will be a random sequence in $\mathbb{Z}/p\mathbb{Z}$, and the **birthday paradox**, see Example 4.6, will imply that after approximately $\sqrt{p}$ random choices two y's, say y_μ and $y_{\mu+\lambda}$, will coincide with probability $\frac{1}{2}$.

To produce a pseudo-random sequence, Pollard suggested a recursion using a polynomial $f \in \mathbb{Z}[x]$. For an initial value $x_0 \in \mathbb{Z}/n\mathbb{Z}$ set $x_{k+1} := f(x_k) \bmod n$. For $y_k := x_k \bmod p$ it still holds that $y_{k+1} \equiv f(y_k) \bmod p$ since $p \mid n$. One often uses the nonlinear polynomial $f := x^2 + c$ with $c \neq 0 \in \mathbb{Z}/n\mathbb{Z}$, typically $c = \pm 1$.

Recall that any recursive sequence in a finite set eventually becomes periodic, giving this method its name ρ:



There are several **cycle-detection algorithms**. The two most prominent ones are **Floyd's tortoise[2] and hare[3] algorithm** and **Brent's algorithm** [Wik17c]. Their goal is to provide the following properties:

1. Avoid too many comparisons.
2. Find the minimal μ and the period length λ.

[2] German: Schildkröte

[3] German: Feldhase

Of course, only the first goal is relevant for us, where the comparison step in the cycle-detection algorithm has to be replaced by the gcd computation: $\gcd(y_{\mu+\lambda} - y_\mu, n)$.

The following version of the Pollard's ρ method is based on Floyd's algorithm:

Algorithm 10.3 (Pollard's ρ method)**.** Given a composite number $n \in \mathbb{N}$ the following algorithm returns a nontrivial factor of n or fail.

```
1. $x := 1, z := 1, d := 1$
2. while $d = 1$ do
    a) $x := f(x)$
    b) $z := f(f(z))$
    c) $d := \gcd(z - x, n)$
    d) if $d = n$ then **return** fail
3. **return** $d$
```

□

The complexity of this method is due to the birthday paradox $\mathcal{O}\left(\sqrt{p}\right) \leq \mathcal{O}\left(n^{\frac{1}{4}}\right)$.

10.3 Fermat's method

Fermat's method for factoring a composite number n tries to write it as the difference of two squares $n = x^2 - y^2$, yielding the factorization $n = (x + y)(x - y)$. Indeed, for a composite *odd* number $n = ab$ such a representation always exists: Setting $x := \frac{a+b}{2}$ and $y := \frac{a-b}{2}$ we recover $a = x + y$ and $b = x - y$.

Example 10.4. Let $n = 7429$. $x = 227$ and $y = 210$ satisfy $x^2 - y^2 = n$ with $x - y = 17$ and $x + y = 437$. Hence $n = 17 \cdot 437$. □

In the following we discuss generalizations of this attempt, also with ideas on how to find x and y with the needed properties.

10.4 Dixon's method

Dixon's method for factoring a composite odd number n is a relaxation of Fermat's method. It is based on the following fact: If x, y are integers with

$$x^2 \equiv y^2 \bmod n \quad \text{and } x \not\equiv \pm y \bmod n$$

then $\gcd(x - y, n)$ (and $\gcd(x + y, n)$) is a nontrivial divisor of n.

Example 10.5. Let $n = 84923$. Taking $x = 20712$ and $y = 16800$ we compute $x^2 - y^2 = 1728 \cdot n$, $x - y = 3912$ (and $x + y = 37512$). Hence $\gcd(x - y, n) = 163$ and $n = 163 \cdot 521$. □

Algorithm 10.6 (Dixon's algorithm)**.** Given a composite number $n \in \mathbb{N}$ the following algorithm returns a nontrivial factor of n or fail.

1. $F := \{p_1, \dots, p_k\} \subset \mathbb{P}$ be a set of k distinct "small" primes, where k is "small".[4] We call F a **factor base**[5].
2. Find $x_1, \dots, x_m \in \mathbb{N}$ ($m > k$) such that $x_i^2 = p_1^{e_{1i}} \cdots p_k^{e_{ki}} \bmod n$.
3. Set $v_i := ((e_{1i}, \dots, e_{ki}) \bmod 2) \in \mathbb{F}_2^k$ for $i = 1, \dots, m$. Solve the $\mathbb{F}_2$-linear system
$$\sum_{i=1}^{m} \varepsilon_i v_i = 0 \in \mathbb{F}_2^k.$$
for the ε_i. If we cannot find a non-trivial solution of this system of equations we need more x_i and go back to step (2).
4. Set $(a_1, \dots, a_k) := \frac{1}{2} \sum_{i=1}^m \varepsilon_i (e_{1i}, \dots, e_{ki}) \in \mathbb{Z}_{\geq 0}^k$.[6] Define
$$x := \prod_{i=1}^{m} x_i^{\varepsilon_i} \quad \text{and} \quad y := p_1^{a_1} \cdots p_k^{a_k}.$$
Then
$$\boxed{x^2} = \prod_{i=1}^{m} x_i^{2\varepsilon_i} \equiv \prod_{i=1}^{m} \left(p_1^{\varepsilon_i e_{1i}} \cdots p_k^{\varepsilon_i e_{ki}}\right) = p_1^{\sum_{i=1}^m \varepsilon_i e_{1i}} \cdots p_k^{\sum_{i=1}^m \varepsilon_i e_{ki}} = p_1^{2a_1} \cdots p_k^{2a_k} = \boxed{y^2} \bmod n.$$
5. If $x \not\equiv y \bmod n$ then **return** $\gcd(x - y, n)$ else **return** fail.

□

Example 10.7. Again let $n = 7429$. Take $F = \{2, 3, 5, 7\}$. We find
$$\begin{aligned} x_1^2 = 87^2 &\equiv 2^2 \quad\;\; \cdot 5 \cdot 7 \bmod 7429 \\ x_2^2 = 88^2 &\equiv \quad\;\; 3^2 \cdot 5 \cdot 7 \bmod 7429. \end{aligned}$$

Note that in this small example two x_i are enough for getting a unique solution to the following system of linear equations.

We have
$$v_1 = (2, 0, 1, 1) = (0, 0, 1, 1) = (0, 2, 1, 1) = v_2 \in \mathbb{F}_2^4.$$
Thus $\varepsilon_1 = \varepsilon_2 = 1$ since $1 \cdot v_1 + 1 \cdot v_2 = 0 \in \mathbb{F}_2^4$. Hence
$$(a_1, \dots, a_4) = \frac{1}{2}(1 \cdot (2, 0, 1, 1) + 1 \cdot (0, 2, 1, 1)) = (1, 1, 1, 1) \in \mathbb{Z}_{\geq 0}^4.$$

Thus, $x = 87 \cdot 88 \equiv 227 \bmod n$ and $y = 2 \cdot 3 \cdot 5 \cdot 7 \equiv 210 \bmod n$. As we saw in Example 10.4 above $\gcd(x - y, n) = \gcd(17, n) = 17$. □

The complexity of Dixon's method is $\mathcal{O}\left(\exp\left(2\sqrt{2}\sqrt{\log n \log\log n}\right)\right)$.

[4] Rough heuristic: $p_k \approx \exp\left(\frac{1}{2}\sqrt{\ln n \ln\ln n}\right)$.

[5] German: Faktorbasis

[6] Not over $\mathbb{F}_2$ but over $\mathbb{Z}$.

10.5 The quadratic sieve

The **quadratic sieve (QS)**[7] of Pomerance is an optimization of Dixon's method. The goal is to find x_i's close to the square root $\sqrt{n}$ such that x_i^2 is B-smooth $\bmod n$ for a "small" bound $B \in \mathbb{N}$ (see Algorithm 10.6, step (2)).

As candidates for these B-smooth x_i^2 consider the quantities

$$Q(a) := (\lfloor \sqrt{n} \rfloor + a)^2 - n \in \mathbb{Z},$$

for a in some **sieve interval** $S := \{-s, \ldots, s\} \subset \mathbb{Z}$ **with width** s.

As $Q(a)$ might be negative a slight modification of Dixon's method turns out to be useful:

Exercise 10.8. Describe a modified version of Dixon's method allowing the factor base F to include -1 (the "sign"). □

By definition, $Q(a)$ is a square $\bmod n$, that is $Q(a) \equiv x^2 \bmod n$ with $x = \lfloor \sqrt{n} \rfloor + a$. The key observation is the following statement.

Proposition 10.9. Let $n \in \mathbb{N}$, let $q \in \mathbb{N} \setminus \{1\}$, and let $x, a \in \mathbb{Z}$.

1. $x^2 \equiv n \bmod q \iff q \mid Q(a)$ for $a = x - \lfloor \sqrt{n} \rfloor$.
2. $q \mid Q(a) \implies q \mid Q(a + kq)$ for all $k \in \mathbb{Z}$.

□

Proof.

1. This follows by the definition of $Q(a)$ as $Q(a) = x^2 - n$ if $a = x - \lfloor \sqrt{n} \rfloor$.
2. More generally, $Q(x + kq) \equiv Q(x) \bmod q$ since computing $\bmod q$ is a ring homomorphism $\mathbb{Z}[x] \twoheadrightarrow \mathbb{Z}/q\mathbb{Z}[x]$. ■

In words: q is a divisor of $Q(a)$ for $a = x - \lfloor \sqrt{n} \rfloor$ iff the equation $x^2 \equiv n \bmod q$ is solvable. And if q is a divisor of $Q(a)$ then it is a divisor of $Q(a + kq)$ for all $k \in \mathbb{Z}$.

For the composite odd number n define

$$\mathbb{P}(n) := \left\{ p \in \mathbb{P} \mid p = 2 \text{ or } \left(\frac{n}{p}\right) = 1 \right\}.$$

This is the set of all primes for which the equation $x^2 \equiv n \bmod p$ is solvable and[8] $p \nmid n$.

Algorithm 10.10. Fix a bound $B \in \mathbb{N}$ and a factor base $F \subset \mathbb{P}(n)$. For a sieve interval $S := \{-s, \ldots, s\} \subset \mathbb{Z}$ the following algorithm returns the list of those $Q(a)$ with $a \in S$ which are B-powersmooth with prime factors in[9] F.

1. Set $L_a := Q(a)$ for all $a \in S$.
2. For all $p \in F$:
   a) Solve[10] the equation $x^2 \equiv n \bmod p$. Hence, by Proposition 10.9, $p \mid Q(a)$ for all $a \in \mathbb{Z}$

[7] German: Sieb

[8] Recall, $\left(\frac{n}{p}\right) = 0$ means that $p \mid n$ — so we have found a prime divisor of n and we are done.

[9] One says, "which factor over F".

[10] Cf. proof of Theorem 7.22, the Tonelli-Shanks algorithm.

with $a \equiv x - \lfloor \sqrt{n} \rfloor \bmod p$.

b) **Sieve**: For all $a \in S$ with $a \equiv \pm x - \lfloor \sqrt{n} \rfloor \bmod p$, where x is a solution of the equation $x^2 \equiv n \bmod p$: Replace L_a by the quotient $\frac{L_a}{p^e}$, where p^e is the maximal power dividing L_a which is $\leq B$.

3. **return** the list of those $Q(a)$ with $a \in S$ for which $L_a = 1$.

□

The complexity of the Quadratic Sieve is $\mathcal{O}\left(\exp\left((1 + \mathcal{O}(1))\sqrt{\log n \log\log n}\right)\right)$.

Remark 10.11. Note that there exists many more general factorization algorithms like the Number Field Sieve. Moreover, there is a wide range of factorization algorithms that are specialized for given situations. These algorithms are discussed in the lectures *Algorithmic* and *Algebraic Number Theory*. See also, for example, [Wik17d] for an overview. □

Chapter 11

Elliptic curves

Besides the usual groups modulo a prime number p one can also use other groups for public key cryptosystems like RSA or ElGamal. Clearly, all chosen groups must satisfy that the DLP is hard. One such group of high interest comes from algebraic geometry, in particular, elliptic curves.

11.1 The projective space

We first define the basic structures to work on in algebraic geometry.

Definition 11.1. Let K be a field.

1. The set
$$\mathbb{A}^n(K) = K^n = \{(x_1, \ldots, x_n) \mid x_i \in K\}$$
is called the **affine space** of dimension n over K. If K is clear from the context then we will simply write $\mathbb{A}^n$ instead.

2. Two distinct points $P, Q \in \mathbb{A}^n(K)$ uniquely determine an **(affine) line**
$$\overline{PQ} := P + K \cdot (Q - P) := \{P + k(Q - P) \mid k \in K\}.$$
containing both of them.

3. The **projective space** of dimension n over K is defined as the set
$$\mathbb{P}^n(K) := \left(K^{n+1} \setminus \{0\}\right) / K^* := \left\{K^* \cdot x \mid x \in K^{n+1} \setminus \{0\}\right\}.$$
Again we write $\mathbb{P}^n$ if the field K is clear from the context.

4. A point P in $\mathbb{P}^n(K)$ can thus be identified with a 1-dimensional subspace of K^{n+1}. More generally, define the **trace** of a subset $Z \subset \mathbb{P}^n(K)$ to be the subset
$$Z^* \subset K^{n+1} = \left\{x \in K^{n+1} \setminus \{0\} \mid K^* \cdot x \in Z\right\} \cup \{0\}.$$
This gives a one-to-one correspondence between subsets of $\mathbb{P}^n(K)$ and those subsets of the underlying vector space K^{n+1} which are unions of 1-dimensional subspaces.

□

Example 11.2. A **(projective) line** in $\mathbb{P}^n(K)$ is the set of all 1-dimensional subspaces of a 2-dimensional subspace $L \leq K^{n+1}$. We identify the projective line with its trace L. Two distinct points $P, Q \in \mathbb{P}^n(K)$ determine a unique projective line $\overline{PQ} := P + Q$ passing through both of them. $P + Q$ is the 2-dimensional span of P, Q, both viewed as 1-dimensional subspaces of K^{n+1}. □

Homogenous coordinates and affine charts

If $x = (x_0, \dots, x_n) \in K^{n+1} \setminus \{0\}$ then for the point $P = K^* \cdot x$ we write

$$P = (x_0 : \ldots : x_n).$$

We call $x_0, \dots, x_n \in K$ the **homogeneous coordinates** of P. They are uniquely determined by P up to a common nonzero factor:

$$(x_0 : \ldots : x_n) = (y_0 : \ldots : y_n) \iff (y_0, \dots, y_n) = k \cdot (x_0, \dots, x_n) \text{ for some } k \in K^*.$$

Example 11.3. Fix a field K.

1. $\mathbb{A}^1 = \{a | a \in K\}$ and $\mathbb{P}^1 = \{(x : y) \mid (x, y) \in K^2 \setminus \{0\}\}$. Identifying $\mathbb{A}^1$ with the affine subspaces $\{(1, y) \mid y \in K\} \subset K^2$ or $\{(x, 1) \mid x \in K\} \subset K^2$ defines two embeddings

$$\begin{aligned}\varphi_0 &: \mathbb{A}^1 \to \mathbb{P}^1, y \mapsto (1 : y),\\ \varphi_1 &: \mathbb{A}^1 \to \mathbb{P}^1, x \mapsto (x : 1).\end{aligned}$$

   These embeddings are also called (standard) **affine charts** of $\mathbb{P}^1$.
   The elements of the image $\varphi_0(\mathbb{A}^1) \subset \mathbb{P}^1$ (resp. $\varphi_1(\mathbb{A}^1) \subset \mathbb{P}^1$) are called **affine points w.r.t. the chart** φ_0 (resp. φ_1). The point $(0 : 1) \in \mathbb{P}^1$, corresponding to the y-axis in K^2, is the only non-affine point w.r.t. φ_0. It is called the **point at infinity**[1] **w.r.t.** φ_0. Analogously for $(1 : 0)$ and φ_1. Summing up:

$$\mathbb{P}^1 = \underbrace{\varphi_0(\mathbb{A}^1)}_{\text{affine points}} \,\dot\cup\, \{\underbrace{(0:1)}_{\text{pt at }\infty}\} = \underbrace{\varphi_1(\mathbb{A}^1)}_{\text{affine points}} \,\dot\cup\, \{\underbrace{(1:0)}_{\text{pt at }\infty}\}.$$

   The partial inverses are given by the "projections"

$$\begin{aligned}\varphi_0^{-1} &: \mathbb{P}^1 \setminus \{(0 : 1)\} \to \mathbb{A}^1, (x : y) \mapsto \frac{y}{x},\\ \varphi_1^{-1} &: \mathbb{P}^1 \setminus \{(1 : 0)\} \to \mathbb{A}^1, (x : y) \mapsto \frac{x}{y}.\end{aligned}$$

2. $\mathbb{A}^2 = \{(a, b) | a, b \in K\}$ and $\mathbb{P}^2 = \{(x : y : z) \mid (x, y, z) \in K^3 \setminus \{0\}\}$. We have three standard charts

$$\begin{aligned}\varphi_0 &: \mathbb{A}^2 \to \mathbb{P}^2, (y, z) \mapsto (1 : y : z),\\ \varphi_1 &: \mathbb{A}^2 \to \mathbb{P}^2, (x, z) \mapsto (x : 1 : z),\\ \varphi_2 &: \mathbb{A}^2 \to \mathbb{P}^2, (x, y) \mapsto (x : y : 1).\end{aligned}$$

[1] German: unendlich ferner Punkt

We will usually identify $\mathbb{A}^2$ with its image under φ_2 and call its elements the **affine points (w.r.t. φ_2)**. The complementary set

$$U := \mathbb{P}^2 \setminus \varphi_2(\mathbb{A}^2) = \{(x : y : 0) \mid (x, y) \in K^2 \setminus \{0\}\} \subset \mathbb{P}^2$$

is a projective line, called the **line at infinity**[2] **(w.r.t. the chart φ_2)**. We will usually refer to φ_2. Visualize in $K^3 = \mathbb{R}^3$.

□

Algebraic sets and homogenization

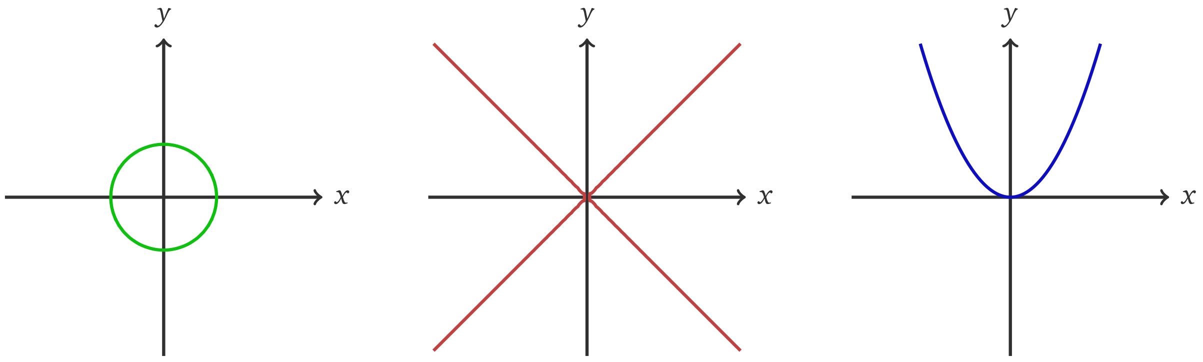
The **vanishing set**

$$V(F) := \{(x, y) \in K^2 \mid F(x, y) = 0\}$$

of *one* polynomial $F \in K[x, y]$ is an example of a so-called **algebraic set**.

Example 11.4. Visualize in $K^2 = \mathbb{R}^2$ the vanishing sets of the degree 2 polynomials

$$F(x, y) = x^2 + y^2 - 1, \quad G(x, y) = x^2 - y^2, \quad \text{and } H(x, y) = x^2 - y.$$



$x^2 + y^2 - 1 = 0$ $x^2 - y^2 = 0$ $x^2 - y = 0$

□

Definition 11.5. Let K be a field.

1. A polynomial $F \in K[x_0, \dots, x_n]$ of degree d is called **homogeneous** if all its monomials are of degree d. It follows that $F(\lambda x_0, \dots, \lambda x_n) = \lambda^d F(x_0, \dots, x_n)$.

2. For a polynomial $F \in K[x, y]$ define the **homogenization** $F^* \in K[x, y, z]$ **(w.r.t. φ_2)** by setting

$$F^*(x, y, z) := z^d F\left(\frac{x}{z}, \frac{y}{z}\right) \in K[x, y, z],$$

where $d = \deg F$. The homogenization is a homogeneous polynomial of degree d.

□

In general, algebraic sets are defined as follows:

[2] German: unendlich ferne Gerade

Definition 11.6. Let K be a field.

1. Let $\mathbb{A}^n = \mathbb{A}^n(K)$ be the affine space over K of dimension n, let $F \in K[x_1, \ldots, x_n]$ be a polynomial over K in n variables. We can interpret F as a K-valued function over $\mathbb{A}^n$ via evaluating F at the points in $\mathbb{A}^n$. Let S be a set of such polynomials F. We define the **affine algebraic set** of S via
$$V(S) := \{x \in \mathbb{A}^n \mid F(x) = 0 \text{ for all } F \in S\} \subset \mathbb{A}^n.$$

2. Let $\mathbb{P}^n = \mathbb{P}^n(K)$ be the projective space over K of dimension n, let $F \in K[x_0, \ldots, x_n]$ be a homogeneous polynomial over K in $n+1$ variables. We can interpret F as a K-valued function over $\mathbb{P}^n$ via evaluating F at the points in $\mathbb{P}^n$. Let S be a set of such polynomials F. We define the **projective algebraic set** of S via
$$V(S) := \{x \in \mathbb{P}^n \mid F(x) = 0 \text{ for all } F \in S\} \subset \mathbb{P}^n.$$

□

Remark 11.7. Let $F \in K[x, y]$. The trace of the image $\varphi_2(V(F))$ coincides with the *affine* points of the vanishing set of the homogenized polynomial F^*:
$$\varphi_2(V(F)) = V(F^*) \setminus U.$$
where $U := \mathbb{P}^2 \setminus \varphi_2(\mathbb{A}^2) = \{(x : y : 0) \mid (x, y) \in K^2 \setminus \{0\}\} \subset \mathbb{P}^2$ (see Example 11.3 (2)). □

Example 11.8. Homogenizing the polynomials in Example 11.4 we get
$$F^*(x, y, z) = x^2 + y^2 - z^2, \quad G^*(x, y, z) = x^2 - y^2, \quad \text{and } H^*(x, y, z) = x^2 - yz.$$
Visualize $V(F^*)$ in $K^3 = \mathbb{R}^3$.

1. $V(x^2 + y^2 - z^2)$ does not intersect the line at infinity $U = \{z = 0\}$ if $K = \mathbb{R}$. What happens for K algebraically closed (e.g., $K = \mathbb{C}$)?
2. $V(x^2 - y^2)$ has exactly two points at infinity, namely $(1 : 1 : 0)$ and $(1 : -1 : 0)$.
3. $V(x^2 - yz)$ meets U in the point $(0 : 1 : 0)$ (but with "multiplicity" 2).

□

In what follows we will often write $F \in K[x, y]$ and mean $V(F^*) \in \mathbb{P}^2$.

Elliptic curves

Let K be a field.

Definition 11.9. The equation
$$E^* : y^2z + a_1xyz + a_3yz^2 = x^3 + a_2x^2z + a_4xz^2 + a_6z^3, \quad a_i \in K.$$
is called the **(homogeneous) Weierstrass equation**. It is the homogenization of the **(affine) Weierstrass equation**
$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in K.$$

Denote the vanishing set of E^* by

$$E(K) := V(E^*) \subset \mathbb{P}^2.$$

□

Remark 11.10. The point $(0 : 1 : 0)$ is the only point of $E(K)$ at infinity, i.e., at $z = 0$. It has "multiplicity" 3 since $E^*(x, y, 0) : x^3 = 0$. □

Remark 11.11 (Normal forms). Depending on the characteristic of the field K one can transform the Weierstrass equation into a simpler form by a coordinate change:

1. If $\operatorname{char} K \neq 2$ then complete the square by substituting $y \to y - \frac{a_1 x + a_3}{2}$ to obtain the normal form

$$y^2 = x^3 + a_2' x^2 + a_4' x + a_6',$$

the right hand side being a cubical univariate polynomial $\left(\text{e.g., } a_2' = a_2 + \frac{a_1^2}{4}\right)$.

2. If $\operatorname{char} K \neq 2, 3$ then the substitution $x \to x - \frac{1}{3} a_2'$ finally yields

$$y^2 = x^3 + ax + b.$$

In the following we will often refer to the normal forms via $f(x) := x^3 + a_2' x^2 + a_4' x + a_6'$ resp. $f(x) := x^3 + ax + b$ depending on the given setting.

□

Singularities

Let K be a field, E a Weierstrass equation, and

$$\begin{aligned} F &:= y^2 + a_1 xy + a_3 y - (x^3 + a_2 x^2 + a_4 x + a_6), \\ F^* &:= y^2 z + a_1 xyz + a_3 yz^2 - (x^3 + a_2 x^2 z + a_4 xz^2 + a_6 z^3) \end{aligned}$$

be the corresponding defining (affine resp. homogeneous) polynomials.

Definition 11.12. Let $P = (x_0 : y_0 : z_0) \in E(K)$.

1. P is called a **singular point (of E)** or simply **singular** if

$$\frac{\partial F^*}{\partial x}(x_0, y_0, z_0) = \frac{\partial F^*}{\partial y}(x_0, y_0, z_0) = \frac{\partial F^*}{\partial z}(x_0, y_0, z_0) = 0.$$

2. $E(K)$ (or E) is called **singular** if there is a singular point $P \in E(K)$, otherwise **nonsingular** or **smooth**.

□

Remark 11.13.

1. $(0 : 1 : 0)$ is not a singular point:

$$\frac{\partial F^*}{\partial z}(0, 1, 0) = (y^2 + a_1 xy + 2a_3 yz - a_2 x^2 - 2a_4 xz - 3a_6 z^2)(0, 1, 0) = 1 \neq 0.$$

2. $\operatorname{char} K \neq 2,3$: $\operatorname{disc}(x^3+ax+b) = -16(4a^3+27b^2)$.
3. $\operatorname{char} K \neq 2$: $E : y^2 = f(x) = x^3 + a_2' x^2 + a_4' x + a_6'$. Then E is singular $\iff$ $\operatorname{disc} f = 0$.

□

Using the notion of singularities we can finally define elliptic curves:

Definition 11.14. E is called an **elliptic curve** if E is smooth. □

For illustrations of elliptic curves see figures 11.1 and 11.2.

Example 11.15. The elliptic curve $E : y^2 = f(x) = x^3 + 2x - 1$ over $K = \mathbb{F}_5$ has 6 affine points plus one point at infinity:

x	0	1	2	3	4
$f(x)$	4	2	1	2	1
y	2, 3	–	1, 4	–	1, 4

$E(\mathbb{F}_5) = \{(0,2),(0,3),(2,1),(2,4),(4,1),(4,4)\} \cup \{\infty\}$. □

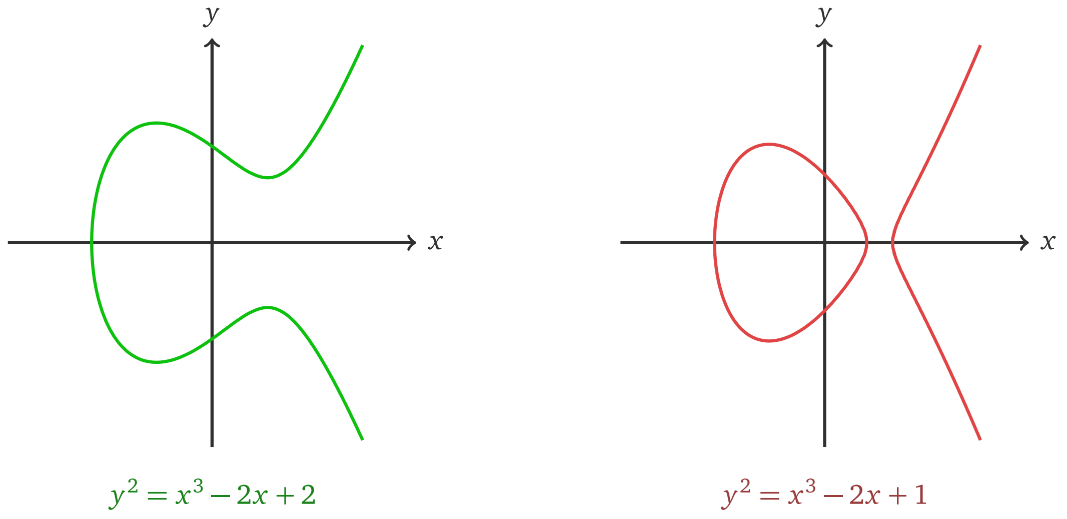


Figure 11.1: Illustration of two elliptic curves given in normal form over $\mathbb{R}$.

11.2 The group structure $(E,+)$

Next we show that there is some group structure on elliptic curves which we want to use in the cryptographic setting.

Let K be a field, $\bar{K}$ its algebraic closure,[3] and E an elliptic curve over K. In this section $\mathbb{P}^2$ refers to $\mathbb{P}^2 := \mathbb{P}^2(\bar{K})$ (the $\bar{K}$ points of E).

[3] For us: Every non-constant polynomial with coefficients in $\bar{K}$ has a root in $\bar{K}$.

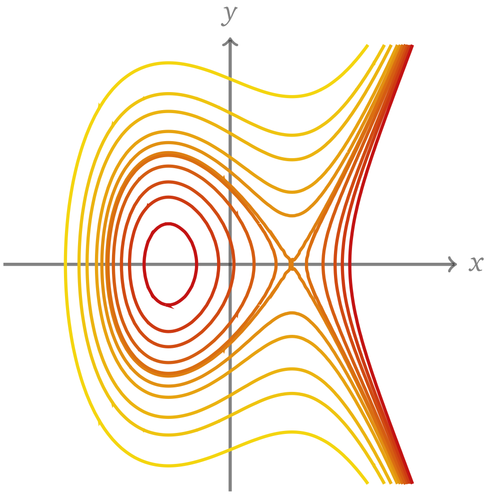


Figure 11.2: A family of elliptic curves: $y^2 = x^3 - 2x + a$ with $-1 < a \leq 6$.

Theorem 11.16. Let $L \subset \mathbb{P}^2$ be a line. Then $|L \cap E(\bar{K})| = 3$, counted with multiplicity. □

Proof. The idea is the following: Substituting a parametrization of the line yields an equation of degree 3 in one indeterminate (parameter the line). This has exactly three roots in $\bar{K}$ counted with multiplicity. It is not obvious how this argument takes care of points at infinity. So we give an elementary proof. Let

$$L = \{(x : y : z) \mid ax + by + cz = 0\} \subset \mathbb{P}^2 \text{ with } (a, b, c) \neq (0, 0, 0).$$

1. $a = b = 0$: $L = \left\{(x : y : 0) \subset \mathbb{P}^2\right\}$ is the line at infinity. To compute $L \cap E(\bar{K})$ set $z = 0$ in E to obtain $x^3 = 0$. The infinite far point $(0 : 1 : 0)$ is a root of multiplicity 3.
2. $a \neq 0$ or $b \neq 0$: $L = \{(x : y : 1) \mid ax + by = -c\} \cup \{(b : -a : 0)\}$.
    a) $b \neq 0$: $(b : -a : 0) \neq (0 : 1 : 0)$, hence $(b : -a : 0) \notin E(\bar{K})$ due to Remark 11.10. Now we compute the affine points by substituting $y = -\frac{ax+c}{b}$ in E to obtain a cubic polynomial in x with 3 roots in $\bar{K}$ (counted with multiplicity).
    b) $b = 0, a \neq 0$: $(0 : 1 : 0) \in E(\bar{K}) \cap L$. To determine the affine points substitute $x = -\frac{c}{a}$ in E and obtain a quadratic polynomial in y that has two roots in $\bar{K}$ (counted with multiplicity). This gives 3 points. ■

Remark 11.17. Bézout's theorem states two curves of degree n and m which do not have a common component intersect in nm points counting multiplicities. The previous Theorem is a special case of Bézout's theorem. Two distinct lines intersecting in exactly one point is another a special case of Bézout's theorem. □

Tangents

Let $F^* = y^2z + a_1xyz + a_3yz^2 - (x^3 + a_2x^2z + a_4xz^2 + a_6z^3)$ and let E be the corresponding elliptic curve.

Definition 11.18. Let $P \in E(\bar{K})$. The line

$$T_P := \left\{ (u : v : w) \in \mathbb{P}^2 \mid \frac{\partial F^*}{\partial x}(P) \cdot u + \frac{\partial F^*}{\partial y}(P) \cdot v + \frac{\partial F^*}{\partial z}(P) \cdot w = 0 \right\}$$

is called the **tangent of** E **at** P. One can rewrite the defining equation as $\nabla F^*(P) \cdot \begin{pmatrix} u \\ v \\ w \end{pmatrix} = 0$, where $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$. □

Remark 11.19.

1. We get the following partial derivations:

$$\begin{aligned} \frac{\partial F^*}{\partial x} &= a_1 yz - 3x^2 - 2a_2 xz - a_4 z^2. \\ \frac{\partial F^*}{\partial y} &= 2yz + a_1 xz + a_3 z^2. \\ \frac{\partial F^*}{\partial z} &= y^2 + a_1 xy + 2a_3 yz - a_2 x^2 - 2a_4 xz - 3a_6 z^2. \end{aligned}$$

2. If $P = (0 : 1 : 0)$ then $\nabla F^*(P) = (0, 0, 1)$. Hence $(0 : 1 : 0) \in T_P = \{(u : v : w) \mid w = 0\} = U$, the line at infinity.

3. If $P = (x : y : 1)$ then

$$\begin{aligned} \frac{\partial F^*}{\partial x}(P) &= a_1 y - 3x^2 - 2a_2 x - a_4. \\ \frac{\partial F^*}{\partial y}(P) &= 2y + a_1 x + a_3. \\ \frac{\partial F^*}{\partial z}(P) &= y^2 + a_1 xy + 2a_3 y - a_2 x^2 - 2a_4 x - 3a_6. \end{aligned}$$

   Verify that $\nabla F^*(P) \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 3F(x, y)$.

From (2) and (3) we deduce that $P \in T_P$ for all $P \in E(\bar{K})$. One can prove that $P \in E(\bar{K})$ is a multiple intersection point of T_P and $E(\bar{K})$ with multiplicity at least 2. We have verified this for the infinite point $(0 : 1 : 0)$ which is an intersection point with multiplicity 3. □

Definition 11.20.

1. Fix $O := (0 : 1 : 0) \in E(\bar{K})$.[4]

2. For $P, Q \in E(\bar{K})$ define $P * Q$ by $E(\bar{K}) \cap L = \{P, Q, P * Q\}$, where

$$L := \begin{cases} \overline{PQ} & \text{if} \quad P \neq Q \\ T_P & \text{if} \quad P = Q \end{cases}.$$

[4] O like origin.

3. Finally define the operation $+$[5] for $P,Q \in E(\bar{K})$ by
$$P + Q := (P * Q) * O.$$

□

Lemma 11.21. Let P,Q,R be points on $E(\bar{K})$. Then

1. $*$ and $+$ are commutative.
2. $(P * Q) * P = Q$.
3. $O * O = O$.
4. Let $L \subset \mathbb{P}^2$ be an arbitrary line, $E(\bar{K}) \cap L = \{P,Q,R\}$, then $(P+Q)+R = O$.
5. $P + O = P$.
6. $P + Q = O \iff P * Q = O$.
7. $+$ is associative.
8. $(E(\bar{K}),+)$ is an Abelian group with neutral element O and $-P = P * O$.
9. $E(K)$ is a subgroup of $E(\bar{K})$.

□

Proof.

1. By construction.
2. Definition of $*$ with $L = \{P,Q,P * Q\}$.
3. Remark 11.19.
4. $(P+Q)+R := ((\underbrace{(P * Q)}_{R} * O) * R) * O \overset{(2)}{=} O * O \overset{(3)}{=} O$.
5. $P + O = (P * O) * O = (O * P) * O \overset{(2)}{=} P$.
6. If $P * Q = O$ then $P + Q = (P * Q) * O = O * O \overset{(2)}{=} O$. Now assume $P + Q = O$. Then
$$P * Q \overset{(5)}{=} (P * Q) + O = ((P * Q) * O) * O = (P + Q) * O = O * O \overset{(2)}{=} O.$$
7. Without a further idea this leads to a lengthy case by case distinction. There exist wonderful geometric[6] ideas to prove the associativity.
8. Follows from (1), (5), (6) and (7)
9. Let E be defined over K and $P,Q \in E(K)$. Then $L, L \cap E$ is defined over K. Moreover, $P * Q$ is, as the third root of $L \cap E(\bar{K})$, also in K. The is a special case of the following simple fact: If $f \in K[x]$ with $\deg f = r$ and if $r-1$ roots are in K then the last root is in K. ■

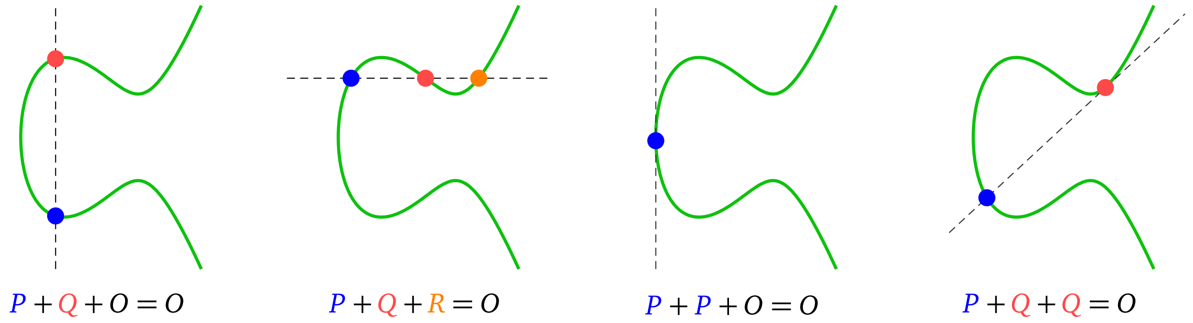


Figure 11.3: The group law on the $\mathbb{R}$-points of the elliptic curve $E : y^2 = x^3 - 2x + 2$

In Figure 11.3 we illustrate the main group actions on E.

Next we give several attempts for computing special values in the elliptic curve group. Recall again: $F := y^2 + a_1xy + a_3y - (x^3 + a_2x^2 + a_4x + a_6)$.

A formula for $-P := P * O$ where $P \neq O$

Let $P = (x_0, y_0) = (x_0 : y_0 : 1)$, so P is affine. We want to determine $\overline{PO}$. The following equivalences are immediate:

$$(x:y:1) \in \overline{PO} \iff \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in \left\langle \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right\rangle \iff x = x_0.$$

So $\overline{PO} = \{(x_0 : y : 1) \mid y \in K\} \cup \{O\}$. Note that we know by construction that y_0 is a root of $F(x_0, y)$ ($P \in E(K)$). It follows that

$$(x_0 : y : 1) \in E(K) \iff F(x_0, y) = 0 \iff y = y_0 \text{ or } y = y_1$$

where

$$F(x_0,y) = (y-y_0)(y-y_1) = y^2 + (-y_0-y_1)y + y_0y_1 \overset{!}{=} y^2 + a_1x_0y + a_3y - (x_0^3 + a_2x_0^2 + a_4x_0 + a_6).$$

Coefficient matching yields: $-y_0 - y_1 = a_1x_0 + a_3$, so $y_1 = -y_0 - a_1x_0 - a_3$. Finally,

$$\boxed{-P = P * O = (x_0 : -y_0 - a_1x_0 - a_3 : 1).}$$

For the most important special case when E is given in normal form we get $a_1 = a_3 = 0$ and

$$\boxed{-(x_0, y_0) = (x_0, -y_0).}$$

[5] Caution: This is different from the sum of traces $P + Q = \overline{PQ}$ from Definition 11.1.

[6] The most elegant proof uses the theory of divisors [Har77, Chapter IV, Section 4].

A formula for $P * Q$ where $P, Q \neq O$

Let $P = (x_1, y_1) = (x_1 : y_1 : 1)$ and $Q = (x_2, y_2) = (x_2 : y_2 : 1)$ be two affine points. By definition,

$$P * Q = O \iff P * O = Q \iff -P = Q.$$

Thus, $x_1 = x_2$ and $y_1 + y_2 + a_1 x_1 + a_3 = 0$. For each line $L \subset \mathbb{P}^2$ we have: $O = (0:1:0) \in L \iff L \cap \mathbb{A}^2$ is parallel to the y-axis. Let w.l.o.g. $P * Q \neq O$ (otherwise $Q = -P$). Set

$$L := \begin{cases} \overline{PQ} & \text{if } P \neq Q, \\ T_P & \text{if } P = Q. \end{cases}$$

Then $L \cap \mathbb{A}^2 = \{(x,y) \mid y = \lambda x + \nu\}$ for some $\lambda, \nu \in K$. We have to distinguish two cases:

1. If $P \neq Q$ then

$$\boxed{\lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ and } \nu = y_1 - \lambda x_1.}$$

2. If $P = Q = (x_1, y_1) = (x_1 : y_1 : 1)$ then

$$T_P \cap \mathbb{A}^2 = \left\{(x,y) \mid \frac{\partial F^*}{\partial x}(P) \cdot x + \frac{\partial F^*}{\partial y}(P) \cdot y + \frac{\partial F^*}{\partial z}(P) \cdot 1 = 0\right\}$$

and

$$\begin{aligned} \frac{\partial F^*}{\partial x}(P) &= a_1 y_1 - 3x_1^2 - 2a_2 x_1 - a_4. \\ \frac{\partial F^*}{\partial y}(P) &= 2y_1 + a_1 x_1 + a_3. \end{aligned}$$

Solving for y we recover the slope[7]

$$\boxed{\lambda = -\frac{\frac{\partial F^*}{\partial x}(P)}{\frac{\partial F^*}{\partial y}(P)} = -\frac{a_1 y_1 - 3x_1^2 - 2a_2 x_1 - a_4}{2y_1 + a_1 x_1 + a_3}.}$$

Further $L \cap E(K) = L \cap \mathbb{A}^2 \cap E(K)$. Again we want $F(x, \lambda x + \nu) = 0 = -(x - x_1)(x - x_2)(x - x_3)$.

$$\begin{aligned} -(x - x_1)(x - x_2)(x - x_3) &= -x^3 + (x_1 + x_2 + x_3)x^2 + \dots \\ &\overset{!}{=} (x + \lambda\nu)^2 + a_1 x(x + \lambda\nu) + a_3(x + \lambda\nu) - (x^3 + a_2 x^2 + a_4 x + a_6) \\ &= -x^3 + \lambda^2 x^2 + a_1 \lambda x^2 - a_2 x^2 + \dots \\ &= -x^3 + \left(\lambda^2 + a_1 \lambda - a_2\right) x^2 + \dots \end{aligned}$$

Coefficient matching at x^2 yields: $x_1 + x_2 + x_3 = \lambda^2 + a_1 \lambda - a_2$. Finally, $P * Q = (x_3, y_3)$ with

$$\boxed{\begin{aligned} x_3 &= \lambda^2 + a_1 \lambda - a_2 - x_1 - x_2, \\ y_3 &= \lambda x_3 + \nu = \lambda(x_3 - x_1) + y_1. \end{aligned}}$$

[7] German: Steigung

A formula for $P+Q$ where $P,Q \neq O$

We now put together the above computations for $P+Q=(P*Q)*O$

$$\boxed{P+Q=(\underbrace{\lambda^2+a_1\lambda-a_2-x_1-x_2}_{=x_3},\underbrace{-y_1+\lambda(x_1-x_3)-a_1x_1-a_3}_{=y_3}).}$$

For the most important special case: $y^2=x^3+ax+b$, i.e., $a_1=a_2=a_3=0$, $a_4=a$, and $a_6=b$. Then

$$\boxed{\lambda=\frac{y_2-y_1}{x_2-x_1} \text{ if } P\neq Q \quad \text{or} \quad \lambda=\frac{3x_1^2+a}{2y_1} \text{ if } P=Q.}$$

Finally,

$$\boxed{\begin{aligned} x_3&=\lambda^2-x_1-x_2,\\ y_3&=-y_1+\lambda(x_1-x_3).\end{aligned}}$$

Example 11.22. We take $K=\mathbb{F}_5\cong\mathbb{Z}/5\mathbb{Z}$ and $y^2=x^3+2x-1$. Verify that

$$E(\mathbb{F}_5)=\{(0,2),(0,3),(2,1),(2,4),(4,1),(4,4)\}\cup\{O\}.$$

1. $P=(0,2) \implies -P=(0,-2)=(0,3)$.
2. $Q=(2,1) \implies P+Q=(2,4)$.
3. $P+P=2P=(4,1)$.
4. $(0,3)+(2,1)=(4,1)$.

□

Remark 11.23. The choice of $O:=(0:1:0)$ was arbitrary but convenient for two reasons:

1. It is the unique non-affine point w.r.t. the fixed coordinate system.
2. It satisfies $O*O=O$.

□

11.3 Elliptic curves over finite fields

In order to use the group structure on elliptic curves in the cryptographical setting, we have to investigate operations on E over finite fields.

Squares in finite fields

Finding square roots in the finite field $\mathbb{F}_q$ is the first step to find $\mathbb{F}_q$-points (i.e., points in $E(\mathbb{F}_q)$) on an elliptic curve E.

Remark 11.24. Let p be a prime and q a power of p. Furthermore, recall the squaring homomorphism $q_n:(\mathbb{Z}/n\mathbb{Z})^*\to(\mathbb{Z}/n\mathbb{Z})^*, x\mapsto x^2 \bmod n$ from Remark 7.5. Now let $k:=|\mathbb{F}_q^*|$.

1. $\ker q_k = \{\pm 1\}$. Hence
$$|(\mathbb{F}_q^*)^2| = \begin{cases} \frac{q-1}{2} & \text{for} \quad q \text{ odd}, \\ q-1 & \text{for} \quad q \text{ even}. \end{cases}$$

2. Define for q odd the **quadratic character**
$$\chi : \mathbb{F}_q^* \to \{\pm 1\} \subset \mathbb{F}_q^*, x \mapsto x^{\frac{q-1}{2}}.$$
Note that $\chi(a) = 1 \iff a \in (\mathbb{F}_q^*)^2$. This follows easily from (1) generalizing Theorem 7.8 by Euler.

□

Algorithm 11.25. Given an *odd* prime power q and an $a \in \mathbb{F}_q^*$ with $\chi(a) = 1$. **Return** $b \in \mathbb{F}_q^*$ with $b^2 = a$ using the Tonelli-Shanks algorithm from the proof of Theorem 7.22, slightly generalized for prime powers. □

Counting points

Let E be an elliptic curve over $\mathbb{F}_q$ and $N := |E(\mathbb{F}_q)|$.

Theorem 11.26 (Hasse-Weil)**.** Let $a := q+1-N$ and α, β be the roots of the quadratic polynomial $x^2 - ax + q$. Then
$$|a| \leq 2\sqrt{q}.$$
Further,
$$|E(\mathbb{F}_{q^m})| = q^m + 1 - (\alpha^m + \beta^m)$$
for all $m \in \mathbb{N}$. □

Proof. A good reference is [Was08, Theorem 4.2]. ■

Remark 11.27. For $N = |E(\mathbb{F}_q)|$ the Hasse-Weil theorem estimates
$$q + 1 - 2\sqrt{q} \leq N \leq q + 1 + 2\sqrt{q}.$$
If q is a *prime* one can show that each natural number in this interval occurs as the order of an elliptic curve $E(\mathbb{F}_q)$. □

Example 11.28. Let $E : y^2 = x^3 + 7x + 1$ be an elliptic curve of $\mathbb{F}_{101}$. It is possible to show that the point $(0,1) \in E(\mathbb{F}_{101})$ has order 116 (see Algorithm 11.31 below), so $N = |E(\mathbb{F}_{101})|$ is a multiple of 116. But the Hasse-Weil theorem says that
$$81 \leq 101 + 1 - 2\sqrt{101} \leq N \leq 101 + 1 + 2\sqrt{101} \leq 123,$$
and the only multiple of 116 in this range is 116. Hence, $E(\mathbb{F}_{101})$ is cyclic of order $N := |E(\mathbb{F}_{101})| = 116$, generated by the point $(0,1)$. □

Example 11.29. With little effort we can determine all the points of $E : y^2 + xy = x^3 + 1$ over the small field $\mathbb{F}_2$:

$$E(\mathbb{F}_2) = \{O, (0,1), (1,0), (1,1)\}.$$

So $N = |E(\mathbb{F}_2)| = 4$. We can thus compute $a := q + 1 - N = -1$ where $q = 2$. If we now go into a field extension, say $\mathbb{F}_{2^{101}}$ we can use the Hasse-Weil theorem to count the number of points: First we compute the roots α, β of $x^2 - (-1)x + 2$ which are $\frac{-1\pm\sqrt{-7}}{2}$. Then we apply the formula from above:

$$\begin{aligned}|E(\mathbb{F}_{2^{101}})| &= 2^{101} + 1 - \left[\left(\frac{-1+\sqrt{-7}}{2}\right)^{101} + \left(\frac{-1-\sqrt{-7}}{2}\right)^{101}\right] \\ &= 2^{101} + 1 - 2969292210605269 \\ &= 2535301200456455833701195805484 \approx 2.5 \cdot 10^{30}.\end{aligned}$$

□

Theorem 11.30 ([Was08, Theorem 4.3]). Let p be a prime and $q = p^n$ and define $N := q + 1 - a$ for some $a \in \mathbb{Z}$ with $|a| \leq 2\sqrt{q}$. Then there is an elliptic curve E defined over $\mathbb{F}_q$ with $|E(\mathbb{F}_q)| = N$ if and only if a satisfies one of the following conditions:

1. $\gcd(a, p) = 1$.
2. n is even and $a = \pm 2\sqrt{q}$.
3. n is even, $p \not\equiv 1 \bmod 3$, and $a = \pm\sqrt{q}$.
4. n is odd, $p = 2$ or $p = 3$, and $a = \pm p^{\frac{n+1}{2}}$.
5. n is even, $p \not\equiv 1 \bmod 4$, and $a = 0$.
6. n is odd and $a = 0$.

□

Let $P \in E(\mathbb{F}_q)$. We want to find the order of P as an element of the group $E(\mathbb{F}_q)$. We know that $NP = O$. Of course we don't know N yet, but we know that $q + 1 - 2\sqrt{q} \leq N \leq q + 1 + 2\sqrt{q}$. One could of course try all values in this interval. This would take $4\sqrt{q}$ steps. The following algorithm is faster and runs in about $4q^{\frac{1}{4}}$ steps:

Algorithm 11.31 (**Baby step, giant step**, [Was08, § 4.3.4]). Given $P \in E(\mathbb{F}_q)$ compute its order.

1. Compute $Q = (q + 1)P$.
2. Choose an integer $m > q^{\frac{1}{4}}$. Compute and save the points jP for $j = 0, \dots, m$ (baby steps).
3. Compute the points

   $$Q + k(2mP) \text{ for } k = -m, \dots, m \text{ (giant steps)}$$

   until $Q + k(2mP) = \pm jP$. Then $MP = O$ with $M := q + 1 + 2mk \mp j$.
4. Factor $M = p_1^{e_1} \cdots p_r^{e_r}$. Compute $(M/p_i)P$ for $i = 1, \dots, r$. If $(M/p_i)P = O$ for some i, replace M with M/p_i and repeat the step until $(M/p_i)P \neq O$ for all i. Then M is the order of P.

To determine $N = |E(\mathbb{F}_q)|$ continue as follows:

5. Repeat the previous steps with randomly chosen points in $E(\mathbb{F}_q)$ until the least common multiple of the element order divides only one integer N in the Hasse-Weil interval. It is then the group order N.

□

Example 11.32. Let E be the elliptic curve $y^2 = x^3 - 10x + 21$ over $\mathbb{F}_{557}$ and $P = (2,3) \in E(\mathbb{F}_{557})$.

1. $Q = 558P = (418,33)$.
2. Let $m = 5 > 557^{\frac{1}{4}}$. The list of jP's ("baby steps") is
$$O,\ (2,3),\ (58,164),\ (44,294),\ (56,339),\ (132,364).$$
3. For $k = 1$ we discover that $Q + k(2mP) = (2,3)$ matches the list for $j = 1$ ("giant step"). Hence $(q + 1 + 2mk - j)P = 567P = O$ and $M = 567$.
4. Factor $567 = 3^4 \cdot 7$. Compute $(567/3)P = 189P = O$. Factor $189 = 3^3 \cdot 7$. Compute $(189/3)P = (38,535) \neq O$ and $(189/7)P = (136,360) \neq O$. Therefore, 189 is the order of P.
5. This suffices to conclude that $|E(\mathbb{F}_{557})| = 567$ as we get from the Hasse-Weil theorem that $511 \approx 557 + 1 - 2\sqrt{557} \leq N \leq 557 + 1 + 2\sqrt{557} \approx 605$.

□

Remark 11.33. There exists an algorithm due to **Schoof** which computes the number of points on an elliptic curves over finite fields $\mathbb{F}_q$ in about $\log^8 q$ steps (cf. [Was08, § 4.5]). □

Finding points

Algorithm 11.34. Let q be a power of an odd prime and $E : y^2 = f(x)$ an elliptic curve of $\mathbb{F}_q$ (cf. Remark 11.11). The following algorithm returns an $\mathbb{F}_q$-point of E:

1. Choose $x \in \mathbb{F}_q$ randomly until $f(x) \in (\mathbb{F}_q)^2$ (test $f(x) = 0$ or $f(x)^{\frac{q-1}{2}} = 1 \in \mathbb{F}_q$).
2. Compute a square root y with $y^2 = f(x)$ using Algorithm 11.25.

□

Remark 11.35. For finding points on elliptic curves over $\mathbb{F}_{2^n}$ see [KMWZ04, Exercise 6.2.2, page 136]. □

The structure of the group $(E,+)$

Theorem 11.36 (Structure Theorem for finitely generated Abelian groups)**.** Let A be a finitely generated Abelian group. Then there exist $r, k \in \mathbb{N}_{>0}$ and $n_1, \ldots, n_k \in \mathbb{N}$ with $n_i \mid n_{i+1}$ such that

$$A \cong \mathbb{Z}^r \times \mathbb{Z}/n_1\mathbb{Z} \times \cdots \mathbb{Z}/n_k\mathbb{Z}.$$

r is called the **rank**[8] of A and the n_i's are called the **determinantal divisors** of A. □

[8] Of course, A is finite if and only if its rank is 0.

Theorem 11.37 (Structure Theorem for elliptic curves over finite fields)**.** There exists natural numbers n_1, n_2 with $n_1 \mid n_2$ such that

$$E(\mathbb{F}_q) \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}.$$

□

The statement includes the case $(n_1, n_2) = (1, n)$ in which case $E(\mathbb{F}_q) \cong \mathbb{Z}/n\mathbb{Z}$. One can sharpen this even further.

Theorem 11.38 (Structure Theorem for elliptic curves over finite fields (refined version))**.** Let p, q, and N be as in Theorem 11.30. Write $N = p^e n_1 n_2$ with $p \nmid n_1 n_2$ and $n_1 \mid n_2$ (possibly $n_1 = 1$). There exists an elliptic curve E over $\mathbb{F}_q$ such that

$$E(\mathbb{F}_q) \cong \mathbb{Z}/p^e\mathbb{Z} \times \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$$

if and only if

1. $n_1 \mid q - 1$ in the cases (1), (3), (4), (5), (6) of Theorem 11.30.
2. $n_1 = n_2$ in case (2) of Theorem 11.30.

These are all groups that occur as $E(\mathbb{F}_q)$. □

Example 11.39. Here are all possible isomorphism types of elliptic curves $E(\mathbb{F}_5)$:

1. Hasse-Weil states that $2 \le N = E(\mathbb{F}_q) \le 10$. This leaves us with the following possibilities (according to Theorem 11.36):

$$\begin{gathered}\mathbb{Z}/2,\ \mathbb{Z}/3,\ \mathbb{Z}/4,\ \mathbb{Z}/2\times\mathbb{Z}/2,\ \mathbb{Z}/5,\ \mathbb{Z}/6 \cong \mathbb{Z}/2\times\mathbb{Z}/3,\ \mathbb{Z}/7,\ \mathbb{Z}/8,\\ \mathbb{Z}/2\times\mathbb{Z}/4,\ \underline{\mathbb{Z}/2\times\mathbb{Z}/2\times\mathbb{Z}/2},\ \mathbb{Z}/9,\ \underline{\mathbb{Z}/3\times\mathbb{Z}/3},\ \mathbb{Z}/10 \cong \mathbb{Z}/2\times\mathbb{Z}/5.\end{gathered}$$

2. The above refined structure Theorem 11.38 rules out the underlined groups.

□

11.4 Lenstra's factorization method

We now come to one of the amazing applications of elliptic curves. It can be viewed as an ingenious variation of Pollard's $p-1$ method (see Section 10.1), but one that comes with an extra degree of freedom: Since it is based on elliptic curves, one can vary the used curve, and even run the algorithm for different curves in parallel.

Let n be the composite number that we want to factorize. Lenstra's method relies on the choice of random elliptic curves E_i over the ring $\mathbb{Z}/n\mathbb{Z}$ with random points on $E_i(\mathbb{Z}/n\mathbb{Z})$ (the group law of elliptic curves over rings is more involved [Was08, § 2.11]). If one starts with the elliptic curve then finding a point involves finding a square root modulo n, which, as we saw in Lemma 7.24, is computationally equivalent to factoring n. To overcome this problem the choice of the curve cannot be independent from the choice of the point:

1. Choose a random element a mod n and a random pair $P = (u, v)$ mod n.

2. *Then* compute $b \equiv v^2 - u^3 - au \bmod n$.
3. The random elliptic curve $y^2 = x^3 + ax + b$ has the $\mathbb{Z}/n\mathbb{Z}$-point $P = (u, v)$.

Algorithm 11.40 (Lenstra)**.** The following algorithm takes a composite number n as its input and returns a factor of n or fail.

1. Choose several[9] random elliptic curves $E_i : y^2 = x^3 + a_i x + b_i$ over $\mathbb{Z}/n\mathbb{Z}$ together with $\mathbb{Z}/n\mathbb{Z}$-points P_i (as above).
2. Choose a bound B ($\approx 10^8$) and compute $\mathrm{lcm}\{1, \ldots, B\}P_i$ (or $(B!)P_i$) on $E_i(\mathbb{Z}/n\mathbb{Z})$ for each i.
3. If step (2) fails because some slope λ does not exist modulo n, due to a non-invertible denominator d, then we have found a factor $\gcd(d, n)$ of n. **Return** this factor.
4. **return** fail.

□

By construction, the complexity of the above algorithm depends on the size of the factor. Lenstra's algorithm is one of the fastest factorization algorithms known.

Remark 11.41.

1. Note that if n is not composite, but a prime, the denominator d of the slope λ is always invertible.
2. One can use Remark 11.27 to explain why this method often yields a nontrivial factor. For details see [Was08, p. 193].
3. The method is very effective in finding prime factors $< 10^{40}$. But in cryptographic applications one uses prime numbers with at least 100 digits. In this range the quadratic sieve (QS) and the number field sieve methods (NFS) outperform Lenstra's method. Nevertheless, it is still useful in intermediate steps of several attacks.

□

Example 11.42 ([Was08, Example 7.1])**.** Let us demonstrate the method to factor $n = 4453$. Choose the elliptic curve $y^2 = x^3 + 10x - 2 \bmod n$ with $\mathbb{Z}/n\mathbb{Z}$-point $P = (1, 3)$. Try to compute $3P$. First compute $2P$. The slope of the tangent at P is

$$\frac{3x^2 + 10}{2y} = \frac{13}{6} \equiv 3713 \bmod 4453.$$

Hence $2P = (x, y)$ with

$$x \equiv 3713^2 - 2 \equiv 4332 \bmod 4453, \text{ and } y \equiv -3713(x - 1) - 3 \equiv 3230 \bmod 4453.$$

To compute $3P$ we add P to $2P$. The slope is

$$\frac{3230 - 3}{4332 - 1} = \frac{3227}{4331}.$$

[9] ... depending on your computing resources.

But 4331 is not invertible modulo n since $\gcd(4331, 4453) = 61 \neq 1$, and we have found a factor of n. This gives the factorization $4453 = 61 \cdot 73$. For the elliptic curve this means that

$$E(\mathbb{Z}/4453\mathbb{Z}) = E(\mathbb{Z}/61\mathbb{Z}) \times E(\mathbb{Z}/73\mathbb{Z})$$

by the Chinese Remainder Theorem.

The method worked since $\mathrm{ord}_{E(\mathbb{Z}/61\mathbb{Z})} P = 3$ while $\mathrm{ord}_{E(\mathbb{Z}/73\mathbb{Z})} P = 64$. The improbable coincidence of these two orders would have produced $0 \bmod n$ as the denominator of the slope and the gcd would have been the trivial one $n = 4453$. □

11.5 Elliptic curves cryptography (ECC)

The hardness of solving the DLP in an Abelian group strongly depends on the way the group is represented. For example $\mathbb{F}_p^* \cong (\mathbb{Z}/(p-1)\mathbb{Z}, +)$. The DLP is hard in the former and trivial in the latter case.

The main usage of elliptic curves in cryptography is to provide an alternative realization of Abelian groups for which all known attacks on the DLP quickly lose their strength (see Chapter 12).

In principle, any cryptosystem or signature scheme which is based on the DLP in $\mathbb{F}_q^*$ can be used with elliptic curves. The ElGamal cryptosystem defined in Definition 8.13 is a good example for this. There is even an elliptic curve analogue of RSA [Was08, § 6.8] that was suggested by Koyama-Maurer-Okamoto-Vanstone.

Still, note that there are classes of "weak" elliptic curves that do not provide good security behaviour. Moreover, the efficient *and* secure implementation of ECC is way harder than for other cryptographic primitives. There is a SafeCurves project that aims to catalog curves that are easy to securely implement. Moreover, these curves are designed publicly, so the chance of introducing a backdoor is quite low.

Remark 11.43.

1. Why use elliptic curves for cryptography? The fastest known algorithms for solving DLP on elliptic curves have a complexity of $\mathcal{O}\left(\sqrt{n}\right)$. This means that the problem is way harder than over finite fields. Where does this come from? Well, let us get at least a small idea via looking at the multiplication tables for $\mathbb{F}_{719}$ and $E(\mathbb{F}_{719})$ with E given in normal form as $y^2 = x^3 + 2 + 1$:

   We can see that whereas there is still some kind of structure for $\mathbb{F}_{719}$ it seems to be nearly random for $E(\mathbb{F}_{719})$. Thus it follows that for a k-bit security one needs elliptic curves over $\mathbb{F}_q$ with $q = 2^{2k}$. For example, if you want to get a 128-bit security for RSA your key size should be 3072-bit. For the same security level using elliptic curves, we only need 256-bit. Thus the keys are way smaller and the cryptosystem can, for example, also be used on very small smart cards.

   The "hardest" ECC scheme broken until today is a 112-bit key for the prime field case and a 109-bit key for the binary field case. The prime field case attack was done in 2009 using over 200 PlayStation 3 consoles running for a bit less than 4 months.

2. In 1999, NIST recommended 15 elliptic curves, most of them over $\mathbb{F}_p$ where p is a Mersenne prime.[10] Modulo these primes reduction can be done by a magnitude faster. Still, these

[10] Recall 31!

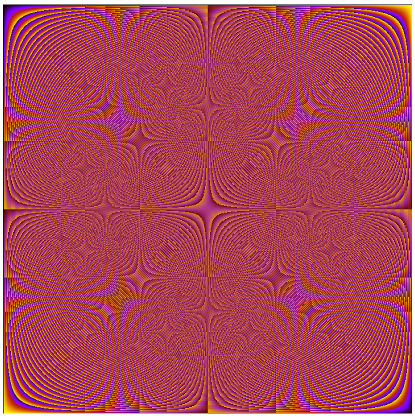


Figure 11.4: $\mathbb{F}_{719}$

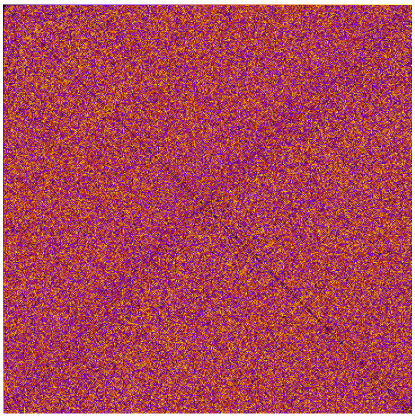


Figure 11.5: $E(\mathbb{F}_{719})$

curves are assumed to have security problems, Bernstein and Lange showed that for some NIST curves the efficiency-related decisions lead to a weakening in the security.

□

A coding function for elliptic curves

The only problem left when using elliptic curves for cryptographic purposes is now: How to encode messages by points on an elliptic curve? The following method is due to Neil Koblitz:

1. Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_p$ with $p \gg 100$ an odd prime and let m be a message encoded as a number $0 \le m < \frac{p}{100}$.
2. For $j = 0, \ldots, 99$ compute $s_j = x_j^3 + ax_j + b$ with $x_j := 100m + j$ and check if s_j is a square (iff $s_j^{\frac{p-1}{2}} \equiv 1 \bmod p$). Stop at the first such j.
3. Computing a square root y_j by Algorithm 11.34 yields an element $P = (x_j, y_j)$ on $E(\mathbb{F}_p)$. Recover the number m encoding the message as $\lfloor \frac{x_j}{100} \rfloor$.
4. Since s_j is a random element of $\mathbb{F}_p^*$ the probability of lying in $(\mathbb{F}_p^*)^2$ is $\frac{1}{2}$. So the probability of failing to find a point P after 100 trials is 2^{-100}.

Chapter 12

Attacks on the discrete logarithm problem

In this chapter we list two attacks on the DLP. One is specific to the group $\mathbb{F}_q^*$ for $q = p^n$ for some prime number p, and the other is independent of the representation of the Abelian group.

12.1 Specific attacks

The index calculus

This attack is specific to the group $\mathbb{F}_q^*$. Here, we only discuss the case $\mathbb{F}_p^*$ where p is an odd prime. The general case $\mathbb{F}_q^*$ requires a bit more work.

So let p be an odd prime and g a generator of the cyclic group $\mathbb{F}_p^*$. The discrete logarithm (cf. Definition 7.2)

$$\log_g : \mathbb{F}_p^* \to \mathbb{Z}/(p-1)\mathbb{Z}$$

defined by $g^{\log_g y} \equiv y \bmod p$ is an isomorphism of cyclic groups, in particular,

$$\log_g(y_1 y_2) \equiv \log_g(y_1) + \log_g(y_2) \bmod (p-1). \tag{12.1}$$

As we mentioned above, the DLP in the source group is hard, while the DLP in the range group is trivial.

The idea of the attack is to compute $\log_g(y)$ for “small” y’s and then to use the identity (12.1) to compute $\log_g$ for arbitrary y’s. Note that

$$\log_g(-1) \equiv \frac{p-1}{2} \bmod (p-1).$$

This is a reformulation of the equation $g^{\frac{p-1}{2}} \equiv -1 \mod p$.

Algorithm 12.1 (Index Calculus). The algorithm takes $y \in \mathbb{F}_p^* = \langle g \rangle$ as input and returns the discrete logarithm $\log_g y \in \mathbb{Z}/(p-1)\mathbb{Z}$ or fail.

1. Choose a bound $B \in \mathbb{N}$ and the factor base $F(B) = \{p \in \mathbb{P} | p \leq B\} \cup \{-1\} = \{-1, p_1, \ldots, p_k\}$.[1]

[1] Recall Dixon’s method in Section 10.4 and the Quadratic Sieve in Section 10.5.

2. Search for y_i's for which the **lift** $b_i \in \mathbb{Z}$ of $g^{y_i} \in \mathbb{F}_p^*$ can be factored over $F(B)$, i.e., $b_i \equiv (-1)^{e_{i0}} \prod_{j=1}^{k} p_j^{e_{ij}} \bmod p$. Do this until the set of equations
$$y_i \equiv \underbrace{\log_g\left((-1)^{e_{i0}}\right)}_{\equiv 0 \text{ or } \frac{p-1}{2}} + e_{i1}\log_g p_1 + \cdots + e_{ik}\log_g p_k \bmod (p-1)$$
can be solved for the vector of k unknowns $(\log_g p_1, \ldots, \log_g p_k)$. If the search fails **return** fail.
3. Search for an ℓ for which the lift $b \in \mathbb{Z}$ of $g^\ell \cdot y \in \mathbb{F}_p^*$ can be factored over $F(B)$, i.e., $b \equiv (-1)^{a_0} \prod_{j=1}^{k} p_j^{a_j} \bmod p$. If the search fails **return** fail.
4. Solve the equation
$$\ell + \log_g y \equiv \underbrace{\log_g\left((-1)^{a_0}\right)}_{\equiv 0 \text{ or } \frac{p-1}{2}} + a_1\log_g p_1 + \cdots + a_k\log_g p_k \bmod (p-1)$$
and **return** $\log_g y$.

□

Example 12.2 ([Was08, Example 5.1]). We will demonstrate the method by computing the discrete logarithm $\log_3 37 \bmod 1217$, so $g = 3$, $y = 37$ and $p = 1217$. Choosing $B = 13$ gives the factor base $F(B) = \{-1, 2, 3, 5, 7, 11, 13\}$. We compute

$$\begin{array}{rclllllll}
3^1 & \equiv & & & 3^1 & & & & \bmod 1217 \\
3^{24} & \equiv & (-1)^1 & \cdot 2^2 & & & \cdot 7 & \cdot 13 & \bmod 1217 \\
3^{25} & \equiv & & & & 5^3 & & & \bmod 1217 \\
3^{30} & \equiv & (-1)^1 & \cdot 2^1 & & \cdot 5^2 & & & \bmod 1217 \\
3^{54} & \equiv & (-1)^1 & & & \cdot 5^1 & \cdot 11 & & \bmod 1217 \\
3^{87} & \equiv & & & & & & 13 & \bmod 1217
\end{array}$$

Recall $p - 1 = 1216$. This yields in particular:

$$\begin{array}{rcrl}
\log_3 2 & \equiv & 216 & \bmod 1216, \\
\log_3 7 & \equiv & 113 & \bmod 1216, \\
\log_3 11 & \equiv & 1059 & \bmod 1216.
\end{array}$$

Finally, for $\ell = 16$, we get $3^{16} \cdot 37 \equiv 2^3 \cdot 7 \cdot 11 \bmod 1217$. Therefore,

$$\log_3 37 \equiv 3\log_3 2 + \log_3 7 + \log_3 11 - 16 \equiv 588 \bmod 1216.$$

□

Remark 12.3. Several remarks on the usage of index calculus and further optimizations:

1. The method was successfully used to compute discrete logarithms modulo a 120-digit prime.
2. Finding the appropriate y_i's and the ℓ can be done using a version of the quadratic sieve (QS) or the number field sieve (NFS) as in Section 10.5.

3. In contrast to $\mathbb{F}_p^* = (\mathbb{Z}/p\mathbb{Z})^*$, elements of $E(\mathbb{F}_p)$ are rarely reductions of elements in $E(\mathbb{Z})$ (or $E(\mathbb{Q})$). It is thus widely believed that the index calculus cannot be adapted to solve the DLP for elliptic curves.

□

12.2 General attacks

By general attacks we mean attacks applicable for all representations of a finite Abelian group. Pollard's ρ method, cf. Section 10.2, can be modified to give a general *probabilistic* algorithm to solve the DLP [Was08, § 5.2.2]. From the many known general attacks on the DLP we only treat a modified version of the baby-step-giant-step algorithm.

Baby Step, Giant Step

Let G be an additively written Abelian group with (known) order N. W.l.o.g. we can assume that $G = \langle P \rangle$.[2]

Algorithm 12.4 (Shanks)**.** The following *deterministic* algorithm takes as input an element $Q \in G$ and returns the discrete logarithm $k := \log_P Q$, i.e., the minimal $k \in \mathbb{N}$ with $kP = Q$.

1. Fix an $m > \sqrt{N}$ and compute mP.
2. Compute and save the list $L := (iP \mid 0 \leq i < m)$ ("baby steps").
3. For $j = 0, \ldots, m-1$ compute the points $Q - jmP$ ("giant steps") until one matches an element in L, say $iP = Q - jmP$.
4. Since $Q = kP$ with $k \equiv i + jm \bmod N$ **return** (the smallest such) k.

□

Proof of correctness. Since $m^2 > N$ it follows that $0 \leq k < m^2$. Write $k = i + jm$ with $0 \leq i < m$ and $0 \leq j = \frac{k-i}{m} < m$. Then $Q - jmP = kP - jmP = iP$. ■

Example 12.5 ([Was08, Example 5.2])**.** Let $G = E(\mathbb{F}_{41})$, where $E : y^2 = x^3 + 2x + 1$. Let $P = (0, 1)$ and $Q = (30, 40)$. The group order N is at most 54 by Hasse-Weil, cf. Theorem 11.26, so set $m = 8$. The list of baby steps iP for $0 \leq i < 8$ consists of the points

$$O,\ (0,1),\ (1,39),\ (8,23),\ (38,38),\ (23,23),\ (20,28),\ (26,9).$$

We start calculating the giant steps for $j = 0, 1, 2, \ldots$

$$(30,40), (9,25), (26,9), \ldots$$

and stop since $(26, 9)$ matches $7P$ in the list. With $i = 7$ and $j = 2$ we compute the discrete logarithm $k = 7 + 2 \cdot 8 = 23$. Indeed: $Q = 23P$. □

[2] We switched notation a bit for the following example applying the algorithm to elliptic curves.

Remark 12.6. The complexity of this attack can be estimated in the following way: We have $\mathcal{O}\left(\sqrt{N}\right)$ computations of the points due to the birthday paradoxon. Now we can assume a hash table to look up if two points match which can be done in $\mathcal{O}(1)$. Thus we get a complexity of $\mathcal{O}\left(\sqrt{N}\right)$ which is much better than general brute force attacks with $\mathcal{O}(N)$. Still, for bigger groups the algorithm is not practical. □

Chapter 13

Digital signatures

For a **digital signature** or **digital signature scheme (DSS)** we understand a mathematical scheme for ensuring the *authenticity* of data. A digital signature should lose its validity if anything in the signed data was altered. This is one of the major advantages compared to ink on paper signatures.

13.1 Basic Definitions & Notations

Definition 13.1. An **asymmetric signature** is a 5-tuple $(\mathscr{P}, \mathscr{C}, \kappa : \mathscr{K}' \to \mathscr{K}, \mathscr{S}, \mathscr{V})$ with the following properties:

1. $\mathscr{P}$ is called the set of **messages**.
2. $\mathscr{C}$ is called the set of **signatures**.
3. $\kappa : \mathscr{K}' \to \mathscr{K}, d \mapsto e$ is a bijective map from the set of **secret signing keys** to the set of **public verification keys**.
4. $\mathscr{S} = (\mathscr{S}_d : \mathscr{P} \rightsquigarrow \mathscr{C})_{d \in \mathscr{K}'}$ is a family of *multi-valued* polynomial algorithms, called the **signing algorithm**.
5. $\mathscr{V} = (\mathscr{V}_e : \mathscr{P} \times \mathscr{C} \to \{0,1\})_{e \in \mathscr{K}}$ is a family of polynomial algorithms, called the **signature verifications** satisfying $\mathscr{V}_{\kappa(d)}(m, \mathscr{S}_d(m)) = 1$ for all $m \in \mathscr{P}$.

□

Usually, one signs only hash values of messages for performance reasons: “hash-then-sign”. We will come to hash functions below.

Definition 13.2. We list the following attack models on a DSS:

1. **Key-only attack**: The attacker only knows the public verification key of the signer.
2. **Known-message attack (KMA)**: The attacker receives some messages (he did not choose them) and their corresponding signatures.
3. **(Adaptive) chosen-message attack (CMA)**: The attacker is allowed to (adaptively) choose messages and receives the corresponding signatures.

□

Definition 13.3. We list the following goals of an attack on a DSS:

1. **Total break**: Recover the signing key.
2. **Universal forgery**:[1] Forge signatures of any message.
3. **Existential forgery**: Forge a signature for some message (without the ability to do this for any message).

□

The strongest security model among the above combinations is the security against universal forgery under an adaptive chosen message attack.

13.2 Signatures using OWF with trapdoors

Let $f := (f_i)_{i\in I} : X \to X$ be a OWF with trapdoor information $(t_i)_{i\in I}$ as in Definition 7.28 (e.g., the Rabin function, cf. Example 7.29 or the RSA function, cf. Example 8.2). Set

1. $\mathscr{P} = \mathscr{C} = X$,
2. $\mathscr{K} = I$, $\mathscr{K}' = \{(i, t_i) \mid i \in I\}$, $\kappa : d := (i, t_i) \mapsto e := i$,
3. $\mathscr{S}_d : \mathscr{P} \to \mathscr{C}$, $m \mapsto f_e^{-1}(m)$ (using the trapdoor information),
4. $\mathscr{V}_e : \mathscr{P} \times \mathscr{C} \to \{0,1\}, (m,s) \mapsto \begin{cases} 1 & \text{if} \quad f_e(s) = m \\ 0 & \text{if} \quad f_e(s) \neq m \end{cases}$.

Remark 13.4.

1. Existential forgery is always possible: First choose the signature s, then choose the message $m := f_e(s)$.
2. If the OWF f is multiplicative (e.g., the RSA function: $(xy)^e = x^e y^e$) then the universal forgery under an adaptive chosen-message attack is possible: To sign m decompose it as $m = m_1 m_2$ with $m_1 \neq m \neq m_2$. Ask for the signatures of s_i of m_i (this is allowed in CMA since $m_i \neq m$). Compute $(m,s) = (m, s_1 s_2)$ by the multiplicativity of f.
3. Another obvious drawback of this scheme is that the signature has the same length as the message.

□

We now give a variant of asymmetric signatures that, under certain assumptions, avoids the above mentioned drawbacks. Recall similar techniques we applied to asymmetric crypto systems in Section 8.4.

Definition 13.5 (Hash-then-sign)**.** This is a variant of Definition 13.1 with the following modifications (we are still using the notation of this section):

1. $\mathscr{P} = \{0,1\}^\bullet$, $\mathscr{C} = X$.

[1] German: Fälschung

2. $\mathscr{H} : \mathscr{P} \to X$ a public map given by a polynomial algorithm, called the **hash function**.
3. $\mathscr{S}_d : \mathscr{P} \to \mathscr{C},\ m \mapsto f_e^{-1}(\mathscr{H}(m))$.
4. $\mathscr{V}_e : \mathscr{P} \times \mathscr{C} \to \{0,1\},\ (m,s) \mapsto \begin{cases} 1 & \text{if} \quad f_e(s) = \mathscr{H}(m) \\ 0 & \text{if} \quad f_e(s) \neq \mathscr{H}(m) \end{cases}$.

□

To avoid the above attack scenarios the hash function $\mathscr{H}$ must be a one-way non-multiplicative function.

13.3 Hash functions

Definition 13.6.

1. A **hash function** is a function $\mathscr{H} : \{0,1\}^\bullet \to \{0,1\}^\ell$ for some fixed $\ell \in \mathbb{N}$ given by a polynomial algorithm.
2. $\mathscr{H}$ is called **collision resistant** if it is infeasible to find distinct x_1, x_2 with $\mathscr{H}(x_1) = \mathscr{H}(x_2)$.

□

Figure 13.1 illustrates the general process of a digital signature using a hash function and a public cryptosystem for signing and verification:

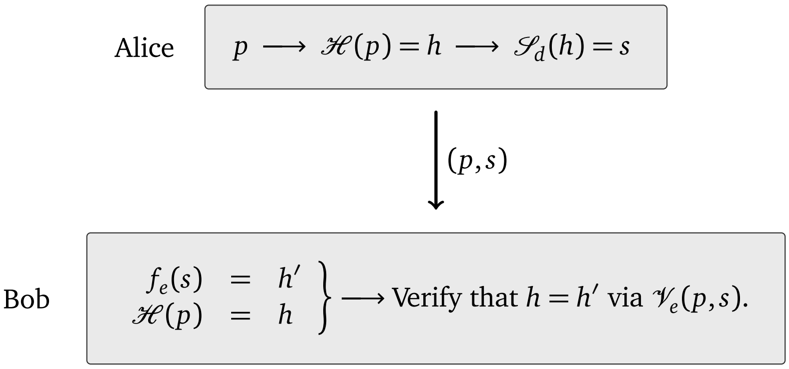

Alice
$p \longrightarrow \mathscr{H}(p) = h \longrightarrow \mathscr{S}_d(h) = s$
(p,s)
Bob
$f_e(s) = h'$
$\mathscr{H}(p) = h$
$\longrightarrow$ Verify that $h = h'$ via $\mathscr{V}_e(p,s)$.

Figure 13.1: Digital signature with hash function and public cryptosystem

Remark 13.7.

1. A collision resistant hash function is a one-way function (since finding a preimage of $\mathscr{H}(x_1)$ would lead to a collision).
2. An “ideal” hash function behaves like a **random oracle (RO)**:
   A random oracle would give to each $x \in \{0,1\}^\bullet$ a *random* answer $\mathscr{H}(x) \in \{0,1\}^\ell$ *and* would store the answer internally as $\mathscr{H}[x]$. If the oracle is given the same x again it will return the cached value $\mathscr{H}[x]$. Note that it is unknown if an “ideal” hash function exists.

3. Note that recently a team of cryptographers found a collision in the SHA-1[2] hash algorithm [Wik17h]. In 2010 the U.S. NIST advised all U.S. federal agencies to no longer use SHA-1 but switch to SHA-2. Still, SHA-1 is used in non-cryptographical applications, for example, the version control systems GIT or Mercurial use this hash algorithm in order to add a SHA-1 checksum to each new commit. In August 2015 the NIST released a new standard called SHA-3 [Wik17i].

□

Example 13.8 (Hash functions from block ciphers)**.** Let $\Sigma = \{0,1\}$, $\ell \in \mathbb{N}_{>0}$, $\mathscr{P} = \Sigma^\ell$, $\mathscr{K} = \Sigma^\ell$ and assume a block cipher such that $\mathscr{E} : \Sigma^\ell \times \mathscr{K} \to \Sigma^\ell, (p,e) \mapsto \mathscr{E}_e(p)$. In particular $\mathscr{E} : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$. Define $\mathscr{H}(x_1,\ldots,x_r) \in \{0,1\}^\ell$ with $x_i \in \Sigma^\ell$ recursively by setting $\mathscr{H}(\emptyset) = 0$ and

$$\mathscr{H}(x_1,\ldots,x_r) = \mathscr{E}_h(x_r) + h, \text{ where } h = \mathscr{H}(x_1,\ldots,x_{r-1}).$$

The widely used SHA-1: $\{0,1\}^\bullet \to \{0,1\}^{160}$ hash function is such an example. The details are too technical. □

Example 13.9. Let p be a prime number such that $q := \frac{p-1}{2}$ is also prime. Further let $b \in \mathbb{F}_p^* = \langle a \rangle$. Define the function

$$f : \{0,\ldots,q-1\} \times \{0,\ldots,q-1\} \to \mathbb{F}_p^*, \ (x,y) \mapsto a^x b^y.$$

As in the previous example, one can use f to construct a hash function. We claim that finding a collision of f implies computing the DL $\log_a b$. □

Proof. Let $a^x b^y = a^{x'} b^{y'}$ be a collision of f, i.e. $(x,y) \neq (x',y')$. If $y = y'$ then $a^x = a^{x'}$ and $x = x'$ ↯ since a generates $\mathbb{F}_p^*$. So let $y \neq y'$. Set $z := \log_a b$. Then

$$a^{x-x'} = b^{y'-y} \implies x - x' \equiv z(y'-y) \bmod (p-1).$$

Now recall that $0 \leq y, y' \leq q-1$, thus $|y'-y| < q$, and $p = 2q$. Thus $\gcd(y'-y, p-1) \in \{1,2\}$. If $\gcd(y'-y,p-1) = 1$ the above formula holds. If $\gcd(y'-y,p-1) = 2$ then we can divide by 2:

$$\frac{x-x'}{2} \equiv \frac{z(y'-y)}{2} \bmod (q).$$

Thus either z or $z+q$ is a solution. ■

13.4 Signatures using OWF without trapdoors

Besides signature schemes whose verification process is based on the existence of a trapdoor function, one can also construct DSS without such an assumption.

[2] SHA stands for Secure Hash Algorithm.

ElGamal signature scheme

Let $G = \langle g \rangle$ be a cyclic group of order N generated by g. Further let $f : G \to \{0,1\}^\bullet$ be a binary representation of the elements of G and $\mathscr{H} : \{0,1\}^\bullet \to \mathbb{Z}/N\mathbb{Z}$ a collision resistant hash function. The **ElGamal signature scheme** is defined by setting:

1. $\mathscr{P} = \{0,1\}^\bullet$.
2. $\mathscr{C} = G \times \mathbb{Z}/N\mathbb{Z}$.
3. $\mathscr{K}' = \{d = (g,a) \mid a \in \mathbb{Z}/N\mathbb{Z}\} \xrightarrow{\kappa} \{e = (g,y) \mid y \in G\} = \mathscr{K},\ (g,a) \mapsto (g, g^a)$.
4. $\mathscr{S}_{(g,a)} : \{0,1\}^\bullet \rightsquigarrow G \times \mathbb{Z}/N\mathbb{Z},\ m \mapsto \sigma$ with σ defined as follows:
   a) Choose randomly $k \in (\mathbb{Z}/N\mathbb{Z})^*$.
   b) Set $r := g^k \in G$.
   c) Set $s := k^{-1}\left(\mathscr{H}(m) + a\mathscr{H}(f(r))\right) \in \mathbb{Z}/N\mathbb{Z}$.
   d) $\sigma := (r,s)$.
5. $\mathscr{V}_{(g,y)} : \{0,1\}^\bullet \times (G \times \mathbb{Z}/N\mathbb{Z}) \to \{0,1\},\ (m,(r,s)) \mapsto \begin{cases} 1 & \text{if } g^{\mathscr{H}(m)} y^{\mathscr{H}(f(r))} = r^s \\ 0 & \text{otherwise} \end{cases}$.

Proof of correctness.

$$r^s = g^{\mathscr{H}(m)} y^{\mathscr{H}(f(r))} \iff ks = \mathscr{H}(m) + a\mathscr{H}(f(r)) \iff s = k^{-1}(\mathscr{H}(m) + a\mathscr{H}(f(r))). \qquad \blacksquare$$

ECDSA

We close this chapter by describing the **elliptic curve** version of the **digital signature algorithm (ECDSA)**. Choose an elliptic curve E over $\mathbb{F}_q$ with $E(\mathbb{F}_q) = \langle P \rangle$ of large prime order N (this assumption can be relaxed, see [Was08, § 6.6]). Choose a *secret* random integer a and compute $Q = aP$ and publish $(E, \mathbb{F}_q, N, P, Q)$. To sign a message with hash value $m \in \mathbb{Z}/N\mathbb{Z}$:

1. Choose a random integer $1 \leq k < N$ and compute $R = kP = (x,y)$.
2. Compute $s = k^{-1}(m + ax) \bmod N$.
3. The signature is (m,R,s).

To verify the signature do the following:

1. Compute $u_1 = s^{-1}m \bmod N$ and $u_2 = s^{-1}x \bmod N$.
2. Compute $V = u_1P + u_2Q$.
3. The signature is valid if $V = R$.

Proof of correctness.

$$V = u_1P + u_2Q = s^{-1}mP + s^{-1}xQ = s^{-1}(mP + xaP) = kP = R. \qquad \blacksquare$$

Appendix A

Some analysis

A.1 Real functions

Jensen's inequality

Lemma A.1. Jensen's inequality Let $f : I \to \mathbb{R}$ be a **strictly concave**, i.e., $\frac{f(x)+f(y)}{2} < f\left(\frac{x+y}{2}\right)$ for all $x, y \in I$ with $x \neq y$). Then for all $a_i > 0$ with $\sum_i a_i = 1$ and all $x_i \in I$ $(i = 1, \dots, n)$

$$\sum_i a_i f(x_i) \leq f\left(\sum_i a_i x_i\right).$$

Equality holds only if $x_1 = \ldots = x_n$. □

The normal distribution

Recall the **normal distribution** $N(\mu, \sigma)$ with expected value μ and variance σ is given by the Gaussian density function

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$N(0, 1)$ is called the **standard** normal distribution.

If X is $N(0, 1)$ distributed and $x > 0$ then

$$\mu_X((-x, x)) = \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right),$$

where

$$\operatorname{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-s^2} ds$$

is the **Gaussian error function**. The function erfc := 1−erf is called the **complementary GAUSSian error function**:

$$\mu_X(\mathbb{R} \setminus (-x, x)) = \operatorname{erfc}\left(\frac{x}{\sqrt{2}}\right).$$

Bibliography

[AGP94] W. R. Alford, Andrew Granville, and Carl Pomerance, *There are infinitely many Carmichael numbers*, Ann. of Math. (2) **139** (1994), no. 3, 703722. MR MR1283874 (95k:11114) 96

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena, *Primes is in p*, vol. 160, June 2004, pp. 781–793. 100

[BH12] Mohamed Barakat and Timo Hanke, *Cryptography – lecture notes*, 2012. i

[Buc04] Johannes Buchmann, *Introduction to cryptography*, 2. ed. ed., Springer, New York [u.a.], 2004. i

[Har77] R. Hartshorne, *Algebraic geometry*, Encyclopaedia of mathematical sciences, Springer, 1977. 118

[KAF$^+$10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thomé, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann, *Factorization of a 768-bit rsa modulus*, Cryptology ePrint Archive, Report 2010/006, 2010, `http://eprint.iacr.org/2010/006`. 89, 103

[KMWZ04] N. Koblitz, A.J. Menezes, Y.H. Wu, and R.J. Zuccherato, *Algebraic aspects of cryptography*, Algorithms and Computation in Mathematics, Springer Berlin Heidelberg, 2004. 123

[Mos17] Stefan Moser, *Information theory – lecture notes*, 2017. 41

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot, *Handbook of applied cryptography*, 1st ed., CRC Press, Inc., Boca Raton, FL, USA, 1996. i

[Sch95] Bruce Schneier, *Applied cryptography (2nd ed.): Protocols, algorithms, and source code in c*, John Wiley & Sons, Inc., New York, NY, USA, 1995. i

[Tre05] Luca Trevisan, *Cs294: Pseudorandomness and combinatorial constructions, lecture notes, lecture 8*, 2005. 68

[Was08] Lawrence C. Washington, *Elliptic curves: Number theory and cryptography, second edition*, 2 ed., Chapman & Hall/CRC, 2008. 121, 122, 123, 124, 125, 126, 129, 130, 136

[Wik16a] Wikipedia, *Berlekampmassey algorithm — wikipedia, the free encyclopedia*, 2016, [Online; accessed 9-March-2017]. 65

[Wik16b] ———, *Drown attack — wikipedia, the free encyclopedia*, 2016, [Online; accessed 14-February-2017]. 2

[Wik16c] ———, *Logjam (computer security) — wikipedia, the free encyclopedia*, 2016, [Online; accessed 14-February-2017]. 2

[Wik16d] ———, *Optimal asymmetric encryption padding — wikipedia, the free encyclopedia*, 2016, [Online; accessed 9-March-2017]. 94

[Wik16e] ———, *Poodle — wikipedia, the free encyclopedia*, 2016, [Online; accessed 14-February-2017]. 2, 18

[Wik16f] ———, *Waldwolfowitz runs test — wikipedia, the free encyclopedia*, 2016, [Online; accessed 9-March-2017]. 64

[Wik17a] ———, *Blum blum shub — wikipedia, the free encyclopedia*, 2017, [Online; accessed 9-March-2017]. 84

[Wik17b] ———, *Computational complexity theory — wikipedia, the free encyclopedia*, 2017, [Online; accessed 16-February-2017]. 7

[Wik17c] ———, *Cycle detection — wikipedia, the free encyclopedia*, 2017, [Online; accessed 16-June-2017]. 104

[Wik17d] ———, *Integer factorization — wikipedia, the free encyclopedia*, 2017, [Online; accessed 16-June-2017]. 108

[Wik17e] ———, *Linear-feedback shift register — wikipedia, the free encyclopedia*, 2017, [Online; accessed 9-March-2017]. 48

[Wik17f] ———, *Mersenne twister — wikipedia, the free encyclopedia*, 2017, [Online; accessed 27-February-2017]. 53

[Wik17g] ———, *Pseudorandom number generator — wikipedia, the free encyclopedia*, 2017, [Online; accessed 9-March-2017]. 48

[Wik17h] ———, *Sha-1 — wikipedia, the free encyclopedia*, 2017, [Online; accessed 3-March-2017]. 135

[Wik17i] ———, *Sha-3 — wikipedia, the free encyclopedia*, 2017, [Online; accessed 3-March-2017]. 135

[Wik17j] ———, *Tonellishanks algorithm — wikipedia, the free encyclopedia*, 2017, [Online; accessed 9-March-2017]. 82

[1]

[1] Wikipedia is a wonderful source of open and easily accessible knowledge. Nevertheless please treat it with care and consult other scientific sources. I cite wikipedia for background material and implementations of widely used algorithms. I do not cite it for theorems or proofs. If you spot a mistake on a page please contribute by correcting it.