Moonmath manual

April 22, 2021

Lorem **ipsum** dolor sit amet, consectetur adipiscing elit. Pellentesque semper viverra dictum. Fusce interdum venenatis leo varius vehicula. Etiam ac massa dolor. Quisque vel massa faucibus, facilisis nulla nec, egestas lectus. Sed orci dui, egestas non felis vel, fringilla pretium odio. *Aliquam* vel consectetur felis. Suspendisse justo massa, maximus eget nisi a, maximus gravida mi.

Here is a citation for demonstration: Lamport et al. [1982]

Chapter 1

Introduction

Nisi vitae suscipit tellus mauris a diam maecenas sed. Amet nisl purus in mollis nunc sed id semper. Et odio pellentesque diam volutpat commodo sed egestas egestas fringilla. Ultricies mi eget mauris pharetra et. Dictum non consectetur a erat nam at lectus urna. Elementum curabitur vitae nunc sed velit. Tincidunt nunc pulvinar sapien et ligula. Turpis cursus in hac habitasse. Hac habitasse platea dictumst quisque sagittis purus. Nam libero justo laoreet sit. Diam ut venenatis tellus in metus vulputate. Lacinia at quis risus sed vulputate. Porta nibh venenatis cras sed felis eget.

Chapter 2

The Zoo of Zero-Knowledge Proofs

First, a list of zero-knowledge proof systems:

1. *Pinocchio (2013): Paper*
   - *Notes: trusted setup*
2. *BCGTV (2013): Paper*
   - *Notes: trusted setup, implementation*
3. *BCTV (2013): Paper*
   - *Notes: trusted setup, implementation*
4. *Groth16 (2016): Paper*
   - *Notes: trusted setup*
   - *Other resources: Talk in 2019 by Georgios Konstantopoulos*
5. *GM17 (207): Paper*
   - *Notes: trusted setup*
   - *Other resources: later Simulation extractability in ROM, 2018*
6. *Bulletproofs (2017): Paper*
   - *Notes: no trusted setup*
   - *Other resources: Polynomial Commitment Scheme on DL, 2016 and KZG10, Polynomial Commitment Scheme on Pairings, 2010*
7. *Ligero (2017): Paper*
   - *Notes: no trusted setup*
   - *Other resources:*
8. *Hyrax (2017): Paper*
   - *Notes: no trusted setup*
   - *Other resources:*

9. *STARKs (2018): Paper*
   - *Notes: no trusted setup*
   - *Other resources:*
10. *Aurora (2018): Paper*
    - *Notes: transparent SNARK*
    - *Other resources:*
11. *Sonic (2019): Paper*
    - *Notes: SNORK - SNARK with universal and updateable trusted setup, PCS-based*
    - *Other resources: Blog post by Mary Maller from 2019 and work on updateable and universal setup from 2018*
12. *Libra (2019): Paper*
    - *Notes: trusted setup*
    - *Other resources:*
13. *Spartan (2019): Paper*
    - *Notes: transparent SNARK*
    - *Other resources:*
14. *PLONK (2019): Paper*
    - *Notes: SNORK, PCS-based*
    - *Other resources: Discussion on Plonk systems and Awesome Plonk list*
15. *Halo (2019): Paper*
    - *Notes: no trusted setup, PCS-based, recursive*
    - *Other resources:*
16. *Marlin (2019): Paper*
    - *Notes: SNORK, PCS-based*
    - *Other resources: Rust Github*
17. *Fractal (2019): Paper*
    - *Notes: Recursive, transparent SNARK*
    - *Other resources:*
18. *SuperSonic (2019): Paper*
    - *Notes: transparent SNARK, PCS-based*
    - *Other resources: Attack on DARK compiler in 2021*

19. *Redshift (2019): Paper*
    - *Notes: SNORK, PCS-based*
    - *Other resources:*

Other resources on the zoo: *Awesome ZKP list on Github, ZKP community with the reference document*

To Do List

- Make table for prover time, verifier time, and proof size
- Think of categories - *Achieved Goals*: Trusted setup or not, Post-quantum or not, ...
- Think of categories - *Mathematical background*: Polynomial commitment scheme, ...
- ... while we discuss the points above, we should also discuss a common notation/language for all these things. (E.g. transparent SNARK/no trusted setup/STARK)

Points to cover while writing

- Make a historical overview over the "discovery" of the different ZKP systems
- Make reader understand what paper is build on what result etc. - the tree of publications!
- Make reader understand the different terminology, e.g. SNARK/SNORK/STARK, PCS, R1CS, updateable, universal, ...
- Make reader understand the mathematical assumptions - and what this means for the zoo.
- Where will the development/evolution go? What are bottlenecks?

Other topics I fell into while compiling this list

- Vector commitments: `https://eprint.iacr.org/2020/527.pdf`
- Snarkl: `http://ace.cs.ohio.edu/~gstewart/papers/snaarkl.pdf`
- Virgo?: `https://people.eecs.berkeley.edu/~kubitron/courses/cs262a-F19/projects/reports/project5_report_ver2.pdf`

Chapter 3

Preliminaries

Introduction and summary of what we do in this chapter

3.1 Cryptological Systems

The science of information security is referred to as *cryptology*. In the broadest sense, it deals with encryption and decryption processes, with digital signatures, identification protocols, cryptographic hash functions, secrets sharing, electronic voting procedures and electronic money. EXPAND

3.2 SNARKS

3.3 complexity theory

Before we deal with the mathematics behind zero knowledge proof systems, we must first clarify what is meant by the runtime of an algorithm or the time complexity of an entire mathematical problem. This is particularly important for us when we analyze the various snark systems...

For the reader who is interested in complexity theory, we recommend, or example or , as well as the references contained therein.

3.3.1 Runtime complexity

The runtime complexity of an algorithm describes, roughly speaking, the amount of elementary computation steps that this algorithm requires in order to solve a problem, depending on the size of the input data.

Of course, the exact amount of arithmetic operations required depends on many factors such as the implementation, the operating system used, the CPU and many more. However, such accuracy is seldom required and is mostly meaningful to consider only the asymptotic computational effort.

In computer science, the runtime of an algorithm is therefore not specified in individual calculation steps, but instead looks for an upper limit which approximates the runtime as soon as the input quantity becomes very large. This can be done using the so-called *Landau notation* (also called big -$\mathcal{O}$-notation) A precise definition would, however, go beyond the scope of this work and we therefore refer the reader to .

For us, only a rough understanding of transit times is important in order to be able to talk about the security of crypographic systems. For example, $\mathcal{O}(n)$ means that the running time of the algorithm to be considered is linearly dependent on the size of the input set n, $\mathcal{O}(n^k)$ means that the running time is polynomial and $\mathcal{O}(2^n)$ stands for an exponential running time (chapter 2.4).

An algorithm which has a running time that is greater than a polynomial is often simply referred to as *slow*.

A generalization of the runtime complexity of an algorithm is the so-called *time complexity of a mathematical problem*, which is defined as the runtime of the fastest possible algorithm that can still solve this problem (chapter 3.1).

Since the time complexity of a mathematical problem is concerned with the runtime analysis of all possible (and thus possibly still undiscovered) algorithms, this is often a very difficult and deep-seated question .

For us, the time complexity of the so-called discrete logarithm problem will be important. This is a problem for which we only know slow algorithms on classical computers at the moment, but for which at the same time we cannot rule out that faster algorithms also exist.

3.4 Software Used in This Book

It order to provide an interactive learning experience, and to allow getting hands-on with the concepts described in this book, we give examples for how to program them in the Sage programming language. Sage is a dialect of the learning-friendly programming language Python, which was extended and optimized for computing with, in and over algebraic objects. Therefore, we recommend installing Sage before diving into the following chapters. The installation steps for various system configurations are described on the sage website[1]. Note however that we use Sage version 9, so if you are using Linux and your package manager only contains version 8, you may need to choose a different installation path, such as using prebuilt binaries.

[1]`https://doc.sagemath.org/html/en/installation/index.html`

Chapter 4

Number Theory

To understand the internals SNARKs it is foremost important to understand the basics of finite field arithmetics AND STUFF.

We therefore start with a brief introduction to fundamental algebraic terms like fields, field extensions AND STUFF . We define these terms in the general abstract way of mathematics, hoping that the non mathematical trained reader will gradually learn to become comfortable with this style. We then give basic examples, that likely all of us know and do basic computations with these examples.

The motivated reader is then encuraged to do some exercises from the apendix of this chapter.

4.1 Preliminaries

INTO-BLA

When you read papers about cryptography or mathematical papers in general, you will frequently stumble across terms like *fields*, *groups*,*rings* and similar. As we will use these terms repeatedly, we have to start this chapter with a short introduction to those terms.

In a nutshell, those terms define sets that are in some sense analog to numbers, in that you can add, subtract, multiply or divide. Or more abstractly those sets have certain ways to combine two elements into a new element.

We know many example like the natural numbers, the integers, the ratinal or the real numbers from school and they are in some sense already the most fundamental examples

Lets start with the concept of a group. Remember back in school when we learned about addition and subtraction of integers. We learned that we can always add to integers and that the result is an integer again. We also learned that nothing happens, when we add zero to any integer, that it doesn't matter in which order we add a given set of integers and that for every integer there is always a negative counterpart and if we add both together we get zero.

Distilling these rules to the smallest list of properties and make rthem abstract we arrive at the definition of a group:

Definition 4.1.0.1 (Group)**.** *A grou $(\mathbb{G}, \cdot)$ is a set $\mathbb{G}$, together with a map $\cdot : \mathbb{G} \cdot \mathbb{G} \to \mathbb{G}$, called the group law (or addition respectively multiplication), such that the following hold:*

- *(Existence of a neutral element) There is a $e \in \mathbb{G}$ for all $g \in \mathbb{G}$, such that $e \cdot g = g$ as well as $g \cdot e = g$.*

- *(Existence of an inverse) For every $g \in \mathbb{G}$ there is a $g^{-1} \in \mathbb{G}$, such that $g \cdot g^{-1} = e$ as well as $g^{-1} \cdot g = e$.*
- *(Associativity) For every $g_1, g_2, g_3 \in \mathbb{G}$ the equation $g_1 \cdot (g_2 \cdot g_3) = (g_1 \cdot g_2) \cdot g_3$ holds.*

In addition a group is called commutative if the equation $g_1 \cdot g_2 = g_2 \cdot g_1$ holds for all $g_1, g_2 \in \mathbb{G}$.

A group is called finite, if the underlying set of elements is finite.

Example 1 (Integer Addition and Subtraction). *As previously described, the set of integers together with integer addition is the archetypical example of a group. In contrast to our definition above the group laws is usually not written as $\cdot$, but as $+$ instead. THe neutral element is the number 0 and the inverse of a natural number $x \in \mathbb{Z}$ is the negative $-x$. Since $x + y = y + x$ this is an example of a commutative group.*

Example 2 (The trivial group). *The most basic example of a group, is group with just one element $\{0\}$ and the group law $0 \cdot 0 = 0$.*

Now thinking of integers again, we know, that there are actually two operations addition and multiplication, such that that they interact in a certain way. This is captured by the concept of a ring, where integers are the most basic example in a sense

Definition 4.1.0.2 (Ring with Unit). *A ring $(R, +, \cdot)$ is a set R, provided with two maps $+ : R \cdot R \to R$ and $\cdot : R \cdot R \to R$, called addition and multiplication, such that the following conditions hold:*

- *$(R, +)$ is a commutative group, where the neutral element is denoted here with 0.*
- *(Existence of a unit) There is an element $1 \in R$ for all $g \in R$,*
- *(Associativity) For every $g_1, g_2, g_3 \in \mathbb{G}$ the equation $g_1 \cdot (g_2 \cdot g_3) = (g_1 \cdot g_2) \cdot g_3$ holds.*
- *(Distributivity) For all $g_1, g_2, g_3 \in R$ the distributive laws apply:*

$$g_1 \cdot (g_2 + g_3) = g_1 \cdot g_2 + g_1 \cdot g_3 \quad \textit{and} \quad (g_1 + g_2) \cdot g_3 = g_1 \cdot g_3 + g_2 \cdot g_3$$

Example 3 (The Ring of Integers). *The set $\mathbb{Z}$ of integers with the usual addition and multiplication is a ring.*

Example 4 (Underlying additive group of a ring). *Every ring $(R, +, \cdot)$ gives rise to group, if we just forget about the multiplication*

The following example is more interesting, as it introduces the concept of so called addition and multiplication table. To understand those tables BLA....

Example 5. *Let $S := \{\bullet, \star, \odot, \otimes\}$ be a set that contains four elements and let adiition and multiplication on S be defined as follows:*

$\cup$	$\bullet$	$\star$	$\odot$	$\otimes$
$\bullet$	$\bullet$	$\star$	$\odot$	$\otimes$
$\star$	$\star$	$\odot$	$\otimes$	$\bullet$
$\odot$	$\odot$	$\otimes$	$\bullet$	$\star$
$\otimes$	$\otimes$	$\bullet$	$\star$	$\odot$

$\circ$	$\bullet$	$\star$	$\odot$	$\otimes$
$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$
$\star$	$\bullet$	$\star$	$\odot$	$\otimes$
$\odot$	$\bullet$	$\odot$	$\bullet$	$\odot$
$\otimes$	$\bullet$	$\otimes$	$\odot$	$\star$

Then $(S, \cup, \circ)$ *is a ring with unit* $\star$ *and zero* $\bullet$*. It therefore makes sense to ask for solutions to equations like this one: Find* $x \in S$ *such that*

$$\otimes \circ (x \cup \odot) = \star$$

To see how such an "alien equation" can be solved, we have to keep in mind, that rings behaves mostly like normal number when it comes to bracketing and computation rules. The only differences are the symbols and the actual way to add and multiply. With this we solve the equation for x *in the "usual way"*

$$\begin{aligned}
\otimes \circ (x \cup \odot) &= \star && \textit{distributive law} \\
\otimes \circ x \cup \otimes \circ \odot &= \star && \otimes \circ \odot = \odot \\
\otimes \circ x \cup \odot &= \star && \textit{add the additive inverse to both sides} \\
\otimes \circ x \cup \odot \cup -\odot &= \star \cup -\odot && \\
\otimes \circ x \cup \bullet &= \star \cup -\odot && \textit{inverse property} \\
\otimes \circ x &= \star \cup -\odot && \textit{neutral element can be deleted} \\
\otimes \circ x &= \star \cup \odot && -\odot = \odot \textit{from addition table} \\
\otimes \circ x &= \otimes && \star \cup \odot = \otimes \textit{from addition table} \\
(\otimes)^{-1} \circ \otimes \circ x &= (\otimes)^{-1} \circ \otimes && \textit{multiply with the multiplicative inverse} \\
\otimes \circ \otimes \circ x &= \otimes \circ \otimes && \\
\star \circ x &= \star && \\
x &= \star &&
\end{aligned}$$

On the other hand, EQUATION $2x = 3$ *has no sultion and equation BLA has more then one solution. So rings are sometimes different from numbers. When we want more we need fields .... BLAHHHH EXPAND*

It is a good and fundamental exercise in mathematics to understand examples like this in detail as it helps the reader to loosen the connection to what "should be" numbers and what not.

Definition 4.1.0.3 (Field)**.** *A field* $(\mathbb{F}, +, \cdot)$ *is a set* $\mathbb{F}$*, provided with two maps* $+ : \mathbb{F} \cdot \mathbb{F} \to \mathbb{F}$ *and* $\cdot : \mathbb{F} \cdot \mathbb{F} \to \mathbb{F}$*, called addition and multiplication, such that the following conditions holds*

- $(\mathbb{F}, +)$ *is a commutative group, where the neutral element is denoted by* 0.
- $(\mathbb{F} \setminus \{0\}, \cdot)$ *is a commutative group, where the neutral element is denoted by* 1.
- *(Distributivity) For all* $g_1, g_2, g_3 \in \mathbb{F}$ *the distributive laws apply:*

$$g_1 \cdot (g_2 + g_3) = g_1 \cdot g_2 + g_1 \cdot g_3 \quad \textit{and} \quad (g_1 + g_2) \cdot g_3 = g_1 \cdot g_3 + g_2 \cdot g_3$$

The characteristic of a field $\mathbb{F}$ *is the smallest natural number* $n \geq 1$*, for which the* n *-fold sum of* 1 *equals zero, i.e. for which* $\sum_{i=1}^{n} 1 = 0$.

If such a $n > 0$ *exists, the field is also called of finite characteristic. If, on the other hand, every finite sum of* 1 *is not equal to zero, then the field is defined to have characteristic* 0.

Example 6 (Field of rational numbers). *Probably the best known example of a field is the set of rational numbers* $\mathbb{Q}$ *together with the usual definition of addition, subtraction, multiplication and division.*

Example 7 (Field with two elements). *It can be shown that in any field, the neutral element* 0 *of addition must be different from the neutral element* 1 *of multiplication, that is we always have* $0 \neq 1$ *in a field. From this follows that the smallest field must contain at least two elements and as the following addition and multiplication tables show, there is indeed a field with two elements, which is usually called* $\mathbb{F}_2$*:*

Let $\mathbb{F}_2 := \{0,1\}$ *be a set that contains two elements and let addition and multiplication on* $\mathbb{F}_2$ *be defined as follows:*

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

4.1.1 Integer Arithmetics

We start by a recapitulation of basic arithmetics as most of us will probably recall from school.

In what follows we write $\mathbb{Z}$ for the set $\{\ldots,-3,-2,-1,0,1,2,3,\ldots\}$ of all integers, together with the usual addition, subtraction and multiplication of integers. Moreover we write$\mathbb{N}$ the set of positive integers and $\mathbb{N}_0$ the set of non-negative integers.

Of course division is not well defined for integers and one of the most basic but at the same time highly important technique, that every reader must become familiar with is the following (kind of) replacement for devision of integers called *Euklidean division* (**?**

Theorem 4.1.1.1 (Euklidean Division). *Let* $a \in \mathbb{Z}$ *be an integer and* $b \in \mathbb{N}$ *a natural number. Then there are always two numbers* $m \in \mathbb{Z}$ *and* $r \in \mathbb{N}$*, with* $0 \leq r < b$ *such that*

$$a = m \cdot b + r \tag{4.1}$$

This decomposition of a given b is called ***Euklidean division*** *or* ***division with remainder*** *and a is called the* ***divident****, b is called the* ***divisor****,m is called the quotient and r is called the* ***remainder****.*

Remark 1. *If* a, b, m *and* r *satisfy the equation (4.1), we often write* $a \ div\ b := m$ *to describe the quotient and use the symbol* $a \bmod b := r$ *for the remainder. We also say, that an integer a is divided by a number b if* $a \bmod b = 0$ *holds. In this case we also write* $a|b$ *(**?** Note 1 below).*

Example 8. *Using our previously defined notation, we have* $-17 \ div\ 4 = -5$ *and* $-17 \bmod 4 = 3$*, because* $-17 = -5 \cdot 4 + 3$ *is the Euklidean division of* -17 *and* 4 *(Since the remainder is by definition a non-negative number). In this case* 4 *does not divide* -17 *as the reminder is not zero. Writing* $-17|4$ *therefore has no meaning.*

On the other hand we can write $12|4$*, since* 4 *divides* 12*, as* $12 \bmod 4 = 0$*.*

A fundamental property of Euklidean division is that both the quotient and the remainder exist and are unique. The result is therefore independent of any algorithm that actually does the computation.

Methods for computing the division with remainder are called *integer division algorithms.* Probably the best known algorithm is the so called *long division*, that most of us might have learned in school. It should be noted however that there are faster algorithms like *Newton–Raphson division* known.

As long division is the standard algorithm used for pen-and-paper division of multi-digit numbers expressed in decimal notation, the reader should become familiar with this algorithm as we use it all over this book when we do simple pen-and-paper computations.

In a nutshell, the algorithm shifts gradually from the left to the right end of the dividend, subtracting the largest possible multiple of the divisor (at the digit level) at each stage; the multiples then become the digits of the quotient, and the final difference is then the remainder. To be more precise one version of the algorithm looks like this:

```
Divide $n$ by $d$
If $d = 0$ then error(DivisionByZeroException) end
$Q \leftarrow 0$ – Initialize quotient to zero
$R \leftarrow 0$ – Initialize remainder to zero
TO APPEAR
```

Example 9 (Integer Log Division). *To give an example of the basic integer long division algorithm most of us learned at school, lets divide the integer* 157843853 *by the number* 261.

So the goal is to find quotient and reminder $m, r \in \mathbb{N}$ such that

$$157843853 = 261 * m + r$$

holds. Using a notation that is mostly used in Commonwealth countries we can then compute

TO-APPEAR

Another important algorithm frequently used in computations with integers is the so-called *extended Euclidean algorithm*, which calculates the greatest common divisor $gcd(a, b)$ of two natural numbers a and $b \in \mathbb{N}$, as well as two additional integers $s, t \in \mathbb{Z}$, such that the equation

$$gcd(a, b) = s \cdot a + t \cdot b \tag{4.2}$$

holds. Two numbers are called **relative prime**, if their greates common divisor is 1.

The following pseudocode shows in detail how to calculate these numbers with the extended Euclidean algorithm (**?** chapter 2.9):

Definition 4.1.1.2. *Let the natural numbers $a, b \in \mathbb{N}$ be given. Then the so-called extended Euclidean algorithm is given by the following calculation rule:*

```
$r_0 := a, \quad r_1 := b, \quad s_0 := 1, \quad s_1 := 0, \quad k := 1$
while $r_k \neq 0$ do
    $q_k := r_{k-1}$ div $r_k$
    $r_{k+1} := r_{k-1} - q_k \cdot r_k$
    $s_{k+1} := s_{k-1} - q_k \cdot s_k$
    $k \leftarrow k + 1$
end while
```

As a result, the algorithm computes the integers $gcd(a, b) := r_k$, as well as $s := s_k$ and $t := (r_k - s_k \cdot a)$ div b such that the equation $gcd(a, b) = s \cdot a + t \cdot b$ holds.

Example 10. *To illustrate the algorithm, lets apply it to the numbers* $(12, 5)$. *We compute*

k	r_k	s_k	$t_k = (r_k - s_k \cdot a)$ div b
0	12	1	0
1	5	0	1
2	2	1	-2
3	1	-2	5

From this one can see that 12 *and* 5 *are relatively prime (since their greatest common divisor is* $gcd(12,5) = 1$*) and that the equation* $1 = (-2) \cdot 12 + 5 \cdot 5$ *holds.*

4.1.2 Modular arithmetic

Congruence or modlar arithmetics (sometimes also called residue class arithmetics) is of central importance for understanding most modern crypto systems. In this section we will therefore take a closer look at this arithmetic. For the notation in cryptology see also **?** Chapter 3, or **?** Chapter 3.

MORE-HIGH-LEVEL-DESCRIPTION

Utilizing Euklidean division as explained previously (4.1.1), congruency of two integers with respect to a so-called moduli can be defined as follows (**?** chapter 3.1):

Definition 4.1.2.1 (congruency)**.** *Let* a, $b \in \mathbb{Z}$ *be two integers and* $n \in \mathbb{N}$ *a natural number. Then* a *and* b *are said to be* ***congruent with respect to the modulus*** n*, if and only if the equation*

$$a \ mod \ n = b \ mod \ n \tag{4.3}$$

holds. In this case we write $a \equiv b \pmod{n}$*. If two numbers are not congruent with respect to a given modulus* n*, we call them incongruent w.r.t.* n*.*

So in other words, if some modulus n is given, then two integers are congruent with respect to this modulus if both Euclidean divisions by n give the same remainder.

Example 11. *To give a simple example, lets assume that we choose the modulus* 271*. Then we have*

$$7 \equiv 2446 \pmod{271}$$

since both $7 \ mod \ 271 = 7$ *as well as* $2446 \ mod \ 271 = 7$

The following theorem describes a fundamental property of modulus arithmetic, which is not known in the traditional arithmetics of integers: (**?** chapter 3.11):

Theorem 4.1.2.2 (Fermat's Little Theorem)**.** *Let* $p \in \mathbb{P}$ *be a prime number and* $k \in \mathbb{Z}$ *is an integer, then:*

$$k^p \equiv k \pmod{p}, \tag{4.4}$$

Proof. □

Remark 2. *Fermats theorem is also often written as the equivalent equation* $k^{p-1} \equiv 1 \pmod{p}$*, which can be derived from the original equation by dividing both sides of the congruency by* k*.*

Another theorem that is important for doing calculations with congruences is the following Chinese remainder theorem, as it provides a way to solves systems congruency equations (**?** chapter 3.15).

Theorem 4.1.2.3 (Chinese remainder theorem)**.** *For any $k \in \mathbb{N}$ let coprime natural numbers $n_1, \ldots n_k \in \mathbb{N}$ as well as the integers $a_1, \ldots a_k \in \mathbb{Z}$ be given. Then the so-called simultaneous congruency*

$$\begin{array}{ll} x \equiv a_1 & (\ mod\ n_1\ ) \\ x \equiv a_2 & (\ mod\ n_2\ ) \\ & \cdots \\ x \equiv a_k & (\ mod\ n_k\ ) \end{array} \tag{4.5}$$

has a solution and all possible solutions of this congruence system are congruent modulo $n_1 \cdot \ldots \cdot n_k$.

Proof. (**?** chapter 3.15) □

Remark 3. *From the proof as given in* **?** *chapter 3.15, the following algorithm to find all solutions to any given system of congruences can be derived TODO:WRITE IN ALGORITHM STYLE*

- *Compute $N = n_1 \cdot n_2 \cdot \ldots \cdot n_k$*
- *For each $1 \leq j \leq k$, compute $N_j = \frac{N}{n_j}$*
- *For each $1 \leq j \leq k$, use the extended Euklidean algorithm (4.1.1.2) to compute numbers s_j as well as t_j, such that $1 = s_j \cdot n_j + t_j \cdot N_j$ holds.*
- *A solution to the congruency system is then given by $x = \sum_{j=1}^{k} a_j \cdot t_j \cdot N_j$.*
- *Compute $m = x\ mod\ N$. The set of all possible solutions is then given by $x + m \cdot \mathbb{Z} = \{\ldots, x-2m, x-m, x, x+m, x+2m, \ldots\}$.*

Remark 4. *This is the classical Chinese remainder theorem as it was already known in ancient China. Under certain circumstances, the theorem can be extended to non-coprime moduli $n_1, \ldots, n_k$.*

Example 12. *To illustrate how to solve simultaneous congruences using the Chinese remainder theorem, let's look at the following system of congruencies:*

$$\begin{array}{ll} x \equiv 4 & (\ mod\ 7\ ) \\ x \equiv 1 & (\ mod\ 3\ ) \\ x \equiv 3 & (\ mod\ 5\ ) \\ x \equiv 0 & (\ mod\ 11\ ) \end{array}$$

So here we have $N = 7 \cdot 3 \cdot 5 \cdot 11 = 1155$, as well as $N_1 = 165$, $N_2 = 385$, $N_3 = 231$ and $N_4 = 105$. From this we calculate with the extended Euclidean algorithm

$$\begin{array}{rrcl} 1 = & -47 \cdot 7 & + & 2 \cdot 165 \\ 1 = & -128 \cdot 3 & + & 1 \cdot 385 \\ 1 = & -46 \cdot 5 & + & 1 \cdot 231 \\ 1 = & -19 \cdot 11 & + & 2 \cdot 105 \end{array}$$

so we have $x = 4 \cdot 2 \cdot 165 + 1 \cdot 1 \cdot 385 + 3 \cdot 1 \cdot 231 + 0 \cdot 2 \cdot 105 = 2398$ as one solution. Because $2398\ mod\ 1155 = 88$ the set of all solutions is $\{\ldots, -2222, -1067, 88, 1243, 2398, \ldots\}$. In particular, there are infinitely many different solutions.

Congruency modulo n is an equivalence relation on the set of integers, where each class is the set of all integers that have the same remainder when divided by n. If then follows from the properties of Euclidean division,that there are exactly n different equivalence classes.

If we go a step further and identify each equivalence class with the corresponding remainder of the Euclidean division, we get a new set, where integer addition and multiplication can be projected to a new kind of addition and multiplication on the equivalence classes.

Roughly speaking the new rules are computed by taking any element of the firsr equivalence class and and element of the second, then add or multiply them in the usual way and see in which equivalence class the result is contained.

The following theorem makes the idea precises

Theorem 4.1.2.4. *Let $n \in \mathbb{N}_{\geq 2}$ be a fixed, natural number and $\mathbb{Z}_n$ the set of equivalence classes of integers with respect to the congruence modulo n relation. Then $\mathbb{Z}_n$ forms a commutative ring with unit with respect to the addition and multiplication defined above.*

Proof. (**?** sentence 1) □

Remark 5. *DESCRIBE NEUTRAL ELEMENTS AND HOW TO ADD, EXPLAIN HOW TO FIND THE NEGATIVE OF A NUMBER AND HOW TO SUBTRACT AND HOW TO MULTIPLY...*

The following example makes the abstract idea more concrete

Example 13 (Arithmetics modulo 6)**.** *Choosing the modulus $n = 6$ we have six equivalence classes of integers which are congruent modulo 6 (which have the same remainder when divided by 6). We write*

$$\begin{array}{lll} 0 := \{\ldots, -6, 0, 6, 12, \ldots\}, & 1 := \{\ldots, -5, 1, 7, 13, \ldots\}, & 2 := \{\ldots, -4, 2, 8, 14, \ldots\} \\ 3 := \{\ldots, -3, 3, 9, 15, \ldots\}, & 4 := \{\ldots, -2, 4, 10, 16, \ldots\}, & 5 := \{\ldots, -1, 5, 11, 17, \ldots\} \end{array}$$

Now to compute the addition of those equivalence classes, say $2 + 5$, one chooses arbitrary elements from both sets say 14 and -1, adds those numbers in the usual way and then looks in which equivalence class the result will be.

So we have $14 + (-1) = 13$ and 13 is in the equivalence class (of) 1. Hence in Z_6 we have that $2 + 5 = 1$!

Applying the same reasoning to all equivalence classes, addition and multiplication can be transferred to the equivalence classes and the results are summarized in the following addition and multiplication tables for $\mathbb{Z}_6$:

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

·	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	2	3	2	1

These two tables are all you need to be able to calculate in $\mathbb{Z}_6$. For example, to determine the multiplicative inverse of a remainder class, look for the entry that results in 1 in the product table. For example the multiplicative inverse of 5 is 5 itself, since $5 \cdot 5 = 1$. Similar to the integers not all numbers have inverses. For example there is no element, that when multiplied with 4 will give 1. However in contrast to what we know from integers, there are non zero numbers, that, when multiplied gives zero (e.g $4 \cdot 4 = 0$).

4.1.3 Polynome

Following **?** we want to give a brief introduction to polynomials.

Definition 4.1.3.1 (Polynomials)**.** *Let R be a commutative ring with unit 1. Then we call the expression*

$$\sum_{n=0}^{m} a_n t^n = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m \ , \tag{4.6}$$

with unknown x and coefficients $a_n \in R$ a ***polynomial with coefficients in*** *R and we write $R[x]$ for the set of all polynomials with coefficients in R.*

We often simply write $P(x) \in R[x]$ for a polynomial and denote the constant term as $P(0)$.

Example 14. *The so-called zero polynomial is the polynomial $0(x) := 0 \cdot x^0$. In analogy to the additively neutral element $0 \in R$ of general rings, we also often write 0 for this polynomial.*

The so-called unit polynomial is the polynomial $1(x) = 1 \cdot x^0$. In analogy to the multiplicatively neutral element $1 \in R$ of a general ring, we also often write 1 for this polynomial.

Definition 4.1.3.2 (degree)**.** *The degree degree$(P(x))$ of a polynomial $P(x) \in R[x]$ is defined as follows: If $P(x)$ is the zero polynomial, we define deg$(P(x)) := -\infty$. For every other polynomial we define degree$(P(x)) = n$ if a_n is the highest non-vanishing coefficient of $P(x)$.*

Polynomials can be added and multiplied in such a way that the set of all polynomials becomes a commutative ring:

Definition 4.1.3.3. *Let $\sum_{n=0}^{\infty} a_n x^n$ and $\sum_{n=0}^{\infty} b_n x$ be two polynomials from $R[x]$. Then the sum and the product of these polynomials is defined as follows:*

$$\sum_{n=0}^{\infty} a_n x^n + \sum_{n=0}^{\infty} b_n x^n = \sum_{n=0}^{\infty} (a_n + b_n) x^n \tag{4.7}$$

$$\left(\sum_{n=0}^{\infty} a_n x^n\right) \cdot \left(\sum_{n=0}^{\infty} b_n x^n\right) = \sum_{n=0}^{\infty} \sum_{i=0}^{n} a_i b_{ni} x^n \tag{4.8}$$

In the case of polynomials, it is only necessary to note that the degree of the sum is exactly the maximum of the degrees of the summands and that the degree of the product is exactly the sum of the degrees of the factors.

The ring of polynomials shares a lot of properties with the integers. In particular there is also the concept of Euclidean division and the algorithm of long division defined for polynomials.

Definition 4.1.3.4 (Irreducible Polynomial)**.** *TECHNOBOB*

4.2 Galois fields

As we have seen in the previous section, modular arithmetics behaves in many ways similar to ordinary arithmetics of integers. But in contrast to arithmetics on integers or rational numbers, we deal with a finite set of elements, which when implemented on computers will not lead to precision problems.

However as we have seen in the last example modular arithmetics is at the same time very different from integer arithmetics as the product of non zero elements can be zero. In addition it is also different from the arithmetics of rational numbers, as there is often no multiplicative inverse, hence no division defined.

In this section we will see that modular arithmetics behaves very nicely, whenever the modulus is a prime number. In that case the rules of modular arithmetics exactly parallels exactly the well know rules of rational arithmetics, despite the fact that the actually computed numbers are very different.

The resulting structures are the so called prime fields and they are the base for many of the contemporary algebra based cryptographic systems.

Since Galois fields are strongly connectedto prime numbers we start with a short overview of prime numbers and provide few basic properties like the fundamental theorem of arithmetic, which says that every natural number can be represented as a finite product of prime numbers.

The key insight here, is that when the modulus is a prime number, modular arithmetic has a well defined division, that is absent for general moduli.

A prime number $p \in \mathbb{N}$ is a natural number $p \geq 2$, which is divisible by itself and by 1 only. Such a prime number is called *odd* if it is not the number 2. We write $\mathbb{P}$ for the set of all prime numbers and $\mathbb{P}_{\geq 3}$ for the set of all odd prime numbers.

As the Greek mathematician Euclid was able to prove by contradiction in the famous *theorem of Euclid*, no largest prime number exists. The set of all prime numbers is thus infinite **?**.

Since prime numbers are especially natural numbers, they can be ordered according to size, so that one can get the sequence

$$p_n := 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, \ldots \tag{4.9}$$

of all prime numbers, which is sequence $A000040$ in OEIS or **?** chapter 1.4). In particular, we can talk about small and large prime numbers.

As the following theorem shows, prime numbers are in a certain sense the basic units from which all other natural numbers are composed:

Theorem 4.2.0.1 (The Fundamental Theorem of Arithmetic)**.** *Let $n \in \mathbb{N}_{\geq 2}$ be a natural number. Then there are prime numbers $p_1, p_2, \ldots, p_k \in \mathbb{P}$, such that:*

$$n = p_1 \cdot p_2 \cdot \ldots \cdot p_k \ . \tag{4.10}$$

*Except for permutations in the factors, this representation is unique and is called the **prime factorization** of n.*

Proof. (**?** sentence 6.) □

Remark 6. *An important question is how fast we can compute the prime factorization of a natural number? This is the famous factorization problem. As far as we know, there* *is no method on a classical Turing machine that is able to compute this representation in polynomial time. The fastest algorithms known today run sub-exponentially, with $\mathcal{O}((1+\epsilon)^n)$ and some $\epsilon > 0$. THe interested reader can find more on this exciting topic in* **?** *Chapter 10.*

Remark 7. *It should be pointed out however hat the American mathematician Peter Williston Shor developed an algorithm in 1994 which can calculate the prime factor representation of a natural number in polynomial time on a quantum computer* **?**.

The consequence of this is, of course, that cryosystems, which are based on the time complexity of the prime factor problem, are unsafe as soon as practically usable quantum computers are available.

Definition 4.2.0.2 (Prime Fields)**.** *(**?** example 3.4.4 or **?** definition 3.1) Let $p \in \mathbb{P}$ be a prime number. Then we write $(\mathbb{F}, +, \cdot)$ for the set of congruency classes and the induced addition and multiplication as described in theorem (**??**) and call it the **prime field** of characteristic p.*

Remark 8. *We have seen in (4.1.2.4) how do compute addition, subtraction and multiplication in modular arithmetics. AS prime fields are just a special case where the modulus is a prime number, all this stays the same. In addition we have also seen in example (XXX) that division is not always possible in modular arithmetics. However the key insight here is, that division is well defined when the modulus is a prime number. This means that in a prime field we can indeed define devision.*

To be more precise, division is really just multiplication with the so called multiplicative inverse, which is really just another element, such that the product of both elements is equal to 1*. This is well known from fractional numbers, where for example the multiplicative element of say* 3 *is simply* 1/3*, since* $3 \cdot 1/3 = 1$*. Division by* 3 *is then noth but multiplication by the inverse* 1/3*. For example* $7/3 = 7 \cdot 1/3$*.*

We can apply the same reasoning when it comes to prime fields and define division as multiplication with the multiplicative inverse, which leads to the question of how to find the multiplicative inverse of an equivalence class $x \in \mathbb{F}_p$ in a prime field.

As with all fields, 0 *has no inverse, which implies, that division by zero is undefined. So lets assume $x \neq 0$. Then $gcd(x,p) = 1$, since p is a prime number and therefore has no divisors (see 4.2.0.1).*

So we can use the extended Euclidean algorithm (REF) to compute numbers $x^{-1}, t \in \mathbb{Z}$ with $s \cdot x + t \cdot p = 1$, which gives x^{-1} as the multiplicative inverse of x in $\mathbb{F}_p$, since $x^{-1}x \equiv 1 \quad (\ mod\ p\ )$.

Example 15. *To summarize the basic aspects of computation in prime fields, lets consider the prime field $\mathbb{F}_5$ and simplify the following expression*

$$\left(\frac{2}{3} - 2\right) \cdot 2$$

A first thing to note is that since F_5 is a field all rules like bracketing (distributivity), summing ect. are identical to the rules we learned in school when we where dealing with rational, real or complex numbers.

So we start by evaluating the bracket and get $\left(\frac{2}{3} - 2\right) \cdot 2 = \frac{2}{3} \cdot 2 - 2 \cdot 2 = \frac{2 \cdot 2}{3} - 2 \cdot 2$. Now we evaluate $2 \cdot 2 = 4$, since $(\ mod\ 4, 5) = 4$ and $-(2 \cdot 2) = -4 = 5 - 4 = 1$, since the negative of a number is just the modulus minus the original number. We therefore get $\frac{2 \cdot 2}{3} - 2 \cdot 2 = \frac{4}{3} + 1$.

Now to compute the faction, we need the multiplicative inverse of 3*, which is the number, that when multiplies with* 3 *in* $\mathbb{F}_5$ *gives* 1*. So we use the extended Euclidean algorithm to compute*

$$x^{-1} \cdot 3 + t \cdot 5 = 1$$

Note that in the Euclidean algorithm the compuations of each t_k *is irrelevant here:*

k	r_k	x_k^{-1}	$t_k = (r_k - s_k \cdot a)$ div b
0	3	1	·
1	5	0	·
2	3	1	·
3	2	-1	·
4	1	2	·

So the multiplicative inverse of 3 *in* $\mathbb{Z}_5$ *is* 2 *and indeed if compute* $3 \cdot 2$ *we get* 1 *in* $\mathbb{F}_5$.

This implies that we can rewrite our original expression into $\frac{4}{3}+1 = 4 \cdot 2+1 = 3+1 = 4$.

The following important property immediately follows from Fermat's little theorem:

Lemma 4.2.0.3. *Let* $p \in \mathbb{P}$ *be a prime number and* $\mathbb{Z}_p$ *be associated prime field of the characteristic* p*. Then the equations*

$$x^p = x \quad or \quad x^{p-1} = 1. \tag{4.11}$$

holds for all elements $x \in \mathbb{Z}_p$ *with* $x \neq 0$*.*

In order to give the reader an impression of how prime fields can be seen, we give a full computation of a prime field in the following example

Example 16 (The prime field $\mathbb{Z}_5$)**.** *For* $n = 5$ *we have five equivalence classes of integers which are congruent modulo* 5*. We write*

$$0 := \{\ldots, -5, 0, 5, 10, \ldots\}, \quad 1 := \{\ldots, -4, 1, 6, 11, \ldots\}, \quad 2 := \{\ldots, -3, 2, 7, 12, \ldots\}$$
$$3 := \{\ldots, -2, 3, 8, 13, \ldots\}, \quad 4 := \{\ldots, -1, 4, 9, 14, \ldots\}$$

Addition and multiplication can now be transferred to the equivalence classes. This results in the following addition and multiplication tables in $\mathbb{Z}_5$*:*

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

These two tables are all you need to be able to calculate in $\mathbb{Z}_5$*. For example, to determine the multiplicative inverse of an element, look for the entry that results in* 1 *in the product table. This is the multiplicative inverse. For example the multiplicative inverse of* 2 *is* 3 *and the multiplicative inverse of* 4 *is* 4 *itself, since* $4 \cdot 4 = 1$*.*

Definition 4.2.0.4 (The finite bodies)**.** *Let* $p \in \mathbb{N}$ *be a prime number. Then* $\mathbb{K}_p$ *denotes the remainder class body örper* $\mathbb{Z}/p\mathbb{Z}$ *and* $\mathbb{K}_{p^n}$*, for each* $n \in \mathbb{N}$*, the finite* K *(unique except for isomorphism) örper with* p^n *elements.*

4.2.1 Square Roots

In this part we deal with *square numbers* and *square roots* in prime fields. To do this, we first define what square roots actually are. We roughly follow Chapter 6.5 in **?**.

Definition 4.2.1.1 (quadratic residue and square roots). *Let $p \in \mathbb{P}$ a prime number and $\mathbb{F}_p$ the associate prime field. Then a number $x \in \mathbb{F}_p$ is called a* ***square root*** *of another number $y \in \mathbb{F}_p$, if x is a solution to the equation*

$$x^2 = y \tag{4.12}$$

On the other hand, if y is given and the quadratic equation has no x solution, we call y as quadratic non-residue. For any $y \in \mathbb{F}_p$ we write

$$\sqrt{y}_{|p} := \{x \in \mathbb{F}_p \mid x^2 = y\} \tag{4.13}$$

for the set of all square roots of y in the prime field $\mathbb{F}_n$. (If y is a quadratic non-residue, then $\sqrt{y}_{|p} = \emptyset$ and if $y = 0$, then $\sqrt{y}_{|p} = \{0\}$)

Remark 9. *The notation $\sqrt{y}_{|n}$ for the root of square residues is not found in textbooks, but it is quite practical to clearly distinguish between roots in different prime fields as the symbol $\sqrt{y}$ is ambiguous and it must also be specified in which prime field this root is actually meant.*

Remark 10. *It can be shown that in any prime field every non zero element has either no square root or two of them. We adopt the convention to call the smaller one (when interpreted as an integer) as the* ***positive*** *square root and the larger one as the* ***negative****. This makes sense, as the larger one can always be computed as the modulus minus the smaller one, which is the definition of the negative in prime fields.*

Example 17 (square numbers and roots in $\mathbb{F}_5$). *Let us consider our example (16) again. All square numbers can be found on the main diagonal of the multiplication table. As you can see, in $\mathbb{Z}_5$ only get the square root of 0, 1 and 4 are non empty sets and we get $\sqrt{0}_{|5} = \{0\}$, $\sqrt{1}_{|5} = \{1,4\}$, $\sqrt{2}_{|5} = \emptyset$, $\sqrt{3}_{|5} = \emptyset$ and $\sqrt{4}_{|5} = \{2,3\}$.*

In order to describe whether an element of a prime field is a square number or not, we define (**?** chapter 6.5) the so called Legendre Symbol as its of used in the literature

Definition 4.2.1.2 (Legendre symbol). *Let $p \in \mathbb{P}$ be a prime number and $y \in \mathbb{F}_p$ an element of the associated prime field. Then the so-called Legendre symbol of y is defined as follows:*

$$\left(\frac{y}{p}\right) := \begin{cases} 1 & \textit{if } y \textit{ has square roots} \\ -1 & \textit{if } y \textit{ has no square roots} \\ 0 & \textit{if } y = 0 \end{cases} \tag{4.14}$$

Example 18. *If we look again at the example (16) we have the following Legendre symbols*

$$\left(\tfrac{0}{5}\right) = 0, \quad \left(\tfrac{1}{5}\right) = 1, \quad \left(\tfrac{2}{5}\right) = -1, \quad \left(\tfrac{3}{5}\right) = -1, \quad \left(\tfrac{4}{5}\right) = 1\ .$$

The following theorem gives a simple criterion for calculating the legendre symbol.

Theorem 4.2.1.3 (Euler criterion)**.** *Let $p \in \mathbb{P}_{\geq 3}$ be an odd Prime number and $y \in \mathbb{F}_p$. Then the Legendre symbol can be computed as*

$$\left(\frac{y}{p}\right) = y^{\frac{p-1}{2}} . \tag{4.15}$$

Proof. (**?** proposition 83) □

So the question remains how to actually compute square roots in prime field. The following algorithms give a solution

Definition 4.2.1.4 (Tonelli-Shanks algorithm)**.** *Let p be an odd prime number $p \in \mathbb{P}_{\geq 3}$ and y a quadratic residue in $\mathbb{Z}_p$. Then the so-called Tonneli* **?** *and Shanks* **?** *algorithm computes the two square roots of y. It is defined as follows:*

1. *Find $Q, S \in \mathbb{Z}$ with $p - 1 = Q \cdot 2^S$ such that Q is odd.*
2. *Find an arbitrary quadratic non-remainder $z \in \mathbb{Z}_p$.*
3.
```
    $M := S, \quad c := z^Q, \quad t := y^Q, \quad R := y^{\frac{Q+1}{2}}, \quad M, c, t, R \in \mathbb{Z}_p$
    while $t \neq 1$ do
        Find the smallest $i$ with $0 < i < M$ and $t^{2^i} = 1$
        $b := c^{2^{M-i-1}}$
         $M := i, \quad c := b^2, \quad t := tb^2, \quad R := R \cdot b$
    end while
```

The results are then the square roots $r_1 := R$ and $r_2 := p - R$ of y in $\mathbb{F}_p$.

Remark 11. *The algorithm (4.2.1.4) works in prime fields for any odd prime numbers. From a practical point of view, however, it is efficient only if the prime number is congruent to 1 modulo 4, since in the other case the formula from the proposition* **??***, which can be calculated more quickly, can be used.*

4.2.2 Exponents and Logarithms

4.2.3 Extension Fields

Eliptic curve pairings often need finite fields that go beyond the finite prime fields.

In fact it is well known, that for every natural number $n \in \mathbb{N}$ there is a field $\mathbb{F}_n$ with n elements, if and only if n is the power of a prime number, i.e. there is a $m \in \mathbb{N}$ with $n = p^m$ for some prime $p \in \mathbb{P}$.

Theorem 4.2.3.1 (Galois Field)**.** *Let $m \in \mathbb{N}$ and $p \in \mathbb{P}$ a prime number. Then there is a field $\mathbb{F}_{p^m}$ of characteristic p, that contains p^n elements.*

We call such a field a **Galois field**. The following algorithm describes the construction of Galois fields (and more general field extensions):

Construction of $\mathbb{F}_{p^m}$

1. Choose a polynomial $P \in \mathbb{F}[t]$ of degree m, that is irreducible.
2. (Field set) of $\mathbb{F}_p$ is the set of all polynomials in $\mathbb{F}[t]$ of degree $< m$, that is

$$\mathbb{F}_{p^m} := \{a_{k-1}t^{k-1} + a_{k-2}t^{k-2} + \ldots + a_1 t + a_0 \mid a_i \in \mathbb{F}_p\}$$

3. (Field Addition) is just addition of the polynomials in the usual way.

4. (Field Multiplication) Multiply the polynomials in the usual way, then compute the long division by P. The remainder is the product.

Remark 12. *In the definition of $\mathbb{F}_{p^m}$, every a_j can have p different values and we have m many of them. This implies that $\mathbb{F}_{p^m}$ contains p^m many elements.*

The construction depends on the choice of an irreducible polynomial, but it can be shown, that all resulting fields are isomorphic, that is they can be transformed into each other, so they are really just different views on the same thing. From an implementations point of view however some choices are better, because they allow for faster computations.

Example 19 (The Galois field $\mathbb{F}_{3^2}$)**.** *In (XXX) we have constructed the prime field $\mathbb{F}_3$. In this example we apply algorithm (XXX) to construct the Galois field $\mathbb{F}_{3^2}$. We start by choosing an irreducibe polynomial of degree 2 with coefficients in $\mathbb{F}_3$. We try $P(t) = t^2+1$. Maybe the fastest way to show that P is indeed irreducible is to just insert all elements from $\mathbb{F}_3$ to see if the result is never zero. WE compute*

$$\begin{aligned} P(0) &= 0^2 + 1 = 1 \\ P(1) &= 1^2 + 1 = 2 \\ P(2) &= 2^2 + 1 = 1 + 1 = 2 \end{aligned}$$

Now the set $\mathbb{F}_{3^2}$ contains all poynomials of degrees lower then two with coefficients in $\mathbb{F}_3$. This is precisely

$$\mathbb{F}_{3^2} = \{0, 1, 2, t, t+1, t+2, 2t, 2t+1, 2t+2\}$$

Addition is then defined as normal addition of polynomials. For example $(t+2)+(2t+2) = (1+2)t + (2+2) = 1$. Doing this computation for all elements give the following addition table

+	0	1	2	t	t+1	t+2	2t	2t+1	2t+2
0	0	1	2	t	t+1	t+2	2t	2t+1	2t+2
1	1	2	0	t+1	t+2	t	2t+1	2t+2	2t
2	2	0	1	r+2	t	t+1	2t+2	2t	2t+1
t	t	t+1	t+2	2t	2t+1	2t+2	0	1	2
t+1	t+1	t+2	t	2t+1	2t+2	2t	1	2	0
t+2	t+2	t	t+1	2t+2	2t	2t+1	2	0	1
2t	2t	2t+1	2t+2	0	1	2	t	t+1	t+2
2t+1	2t+1	2t+2	2t	1	2	0	t+1	t+2	t
2t+2	2t+2	2t	2t+1	2	0	1	t+2	t	t+1

From this table, we can deduce the negative of any element from $\mathbb{F}_{3^2}$. For example in $\mathbb{F}_{3^2}$ we have $-(2t+1) = t+2$, since $(2t+1)+(t+2) = 0$ and the negative of an element is that other element, such that the sum gives the additive neutral element.

Multiplication then needs a bit more computation, as you multiply the polynomials and then divide the result by P and keep the remainder. For example $(t+2)\cdot(2t+2) = 2t^2 + 2t + t + 1 = 2t^2 + 1$. Long division by $P(t)$ then gives $2t^2 + 1 : t^2 + 1 = 2 + \frac{2}{t^2+1}$, so the remainder is 2 and find that the product of $t+2$ and $2t+2$ in $\mathbb{F}_{3^2}$ is 2. Doing this computation for all elements give the following multiplication table (DOTHIS!!! THE TABLE NEEDS AN UPDATE)

·	0	1	2	t	t+1	t+2	2t	2t+1	2t+2
0	0	1	2	t	t+1	t+2	2t	2t+1	2t+2
1	1	2	0	t+1	t+2	t	2t+1	2t+2	2t
2	2	0	1	r+2	t	t+1	2t+2	2t	2t+1
t	t	t+1	t+2	2t	2t+1	2t+2	0	1	2
t+1	t+1	t+2	t	2t+1	2t+2	2t	1	2	0
t+2	t+2	t	t+1	2t+2	2t	2t+1	2	0	1
2t	2t	2t+1	2t+2	0	1	2	t	t+1	t+2
2t+1	2t+1	2t+2	2t	1	2	0	t+1	t+2	t
2t+2	2t+2	2t	2t+1	2	0	1	t+2	t	t+1

From this table, we can deduce the negative of any element from $\mathbb{F}_{3^2}$*. For example in* $\mathbb{F}_{3^2}$ *we have* $-(2t+1) = t+2$*, since* $(2t+1)+(t+2) = 0$ *and the negative of an element is that other element, such that the sum gives the additive neutral element.*

4.3 Epileptic Curves

In this section we introduce epileptic curves as they are used in cryptography, hwhich are certain types of commutative groups basically, well suited for various constructions of various cryptographic primitives.

The eliptic curves we consider are all defined over Galoise fields, so the reader should be familiar with the contend of the previous section.

Definition 4.3.0.1 (Short Weierstraß elliptic Curve). *Let* $\mathbb{F}_q$ *be a Galois field and* $a, b \in \mathbb{F}_q$ *be two field elements. Then a short Weierstrass elliptic curve* E/F_q *over* F_q *is defined as the set*

$$E/\mathbb{F}_q = \{(x,y) \in \mathbb{F}_q \times \mathbb{F}_q \mid y^2 = x^3 + ax + b\} \bigcup \{\mathcal{O}\}$$

of all pairs $(x,y) \in \mathbb{F}_q \times \mathbb{F}_q$ *of field elements, that satisfy the short Weierstrass equation* $y^2 = x^3 + ax + b$*, together with the "point at infinity* $\mathcal{O}$*.*

If the characteristic of the Galois field is 2 or 3, that is if $\mathbb{F}_q$ is of the form $\mathbb{F}_{2^n}$ or $\mathbb{F}_{3^n}$ for some $n \in \mathbb{N}$, then this is not the most general way to describe an elliptic curve. However for our purposes this is al we need.

Example 20. *Lets consider our prime field* F_3 *from (XXX). If we choose* $a = 1$ *and* $b = 0$ *then the corresponding elliptic curve* $E/\mathbb{F}_3$ *is given by all pairs* (x,y) *from* $\mathbb{F}_3$ *such that* $y^2 = x^3 + x$*. We can find this set simply by trying all nine combinations of pairs. We get*

$$E/\mathbb{F}_3 = \{(0,0), (1,2), (2,2), \mathcal{O}\}$$

So our elliptic curve contains four elements.

4.4 Short Weier

Chapter 5

Exercises and Solutions

TODO: All exercises we provided should have a solution, which we give here in all detail.

Bibliography

Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, pages 382–401, July 1982. URL `https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/`.