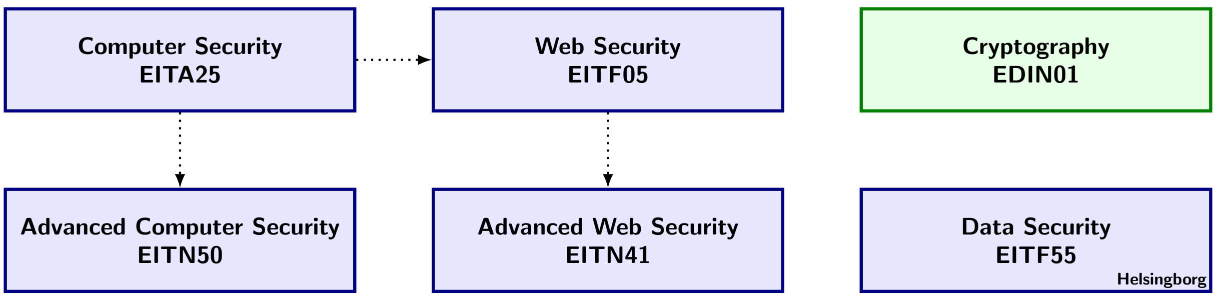
EDIN01 Cryptography 2018

Project 2: Shift Register Sequences

- This project will be done in groups of 1-2 people.
- Estimated effort: 20 man-hours

Learning goals:

- Learn about linear shift register sequences.
- Learn about be Bruijn sequences.
- Implementing devices that output shift register sequences.
- Improving presentation skills.



1 Introduction

The purpose of this project is to learn more about shift register sequences. This will include linear feedback shift register sequences as well as de Bruijn sequences (also called full length sequences). The project covers chapters 1 and 4 in the course literature (lecture notes).

The exercises marked **Home exercises** (HE) should be done by hand. If you run into problems make sure you get some help before you proceed.

There are additional hints and comments on the course home page. In particular, there is a general instruction on factoring polynomials in finite fields, together with an explanation of HE 1.3.

To solve most of the exercises marked **Laboratory exercises** (LE) you need a computer program that can handle symbolic algebra. We have two suggested programs in this course.

1.1 Maple

The first suggestion is `Maple`. The maple commands you will use are e.g., `Factor`, `Primitive`, `Irreduc`, and the `GF` package with `G:-ConvertIn` and `G:-order`. Note that capitalization matters for these commands, `irreduc` and `Irreduc` are two different commands. **Read the help on all these commands before you start the laboratory exercises.**

Students with an LU account can download Maple for free from here.

1.2 SageMath

If you don't want to use Maple, an open-source alternative with simpler syntax is `SageMath`. The easiest way to use SageMath is to register an account at CoCalc. You can also download Sage from here and run it on your own computer. For this project you will only need to use the `SageMath Notebook`, which works in a way similar to `Maple`.

For an introduction to `SageMath` you can make a simple search on the internet to find lots of tutorials. Some useful sage commands for this project are `GF`, `factor`, `is_primitive`, `is_irreducible` and `multiplicative_order`.

When finished with the project, you get your exercises approved by the assistant. You will also have to answer some basic questions to show that you understand what you have done.

2 Polynomials over finite fields

We will start to study whether a certain polynomial is *primitive, irreducible, or reducible* (sv. primitivt, primt, faktoriserbart). If you do not recall the definitions, go back to the lecture notes and study it again.

Home exercise 1: For each of the polynomials below, determine whether the polynomial is primitive, irreducible, or reducible over the specified field.

1. $p(x) = x^4 + x^2 + 1$ over $\mathbb{F}_2$.
2. $p(x) = x^3 + x + 1$ over $\mathbb{F}_3$.
3. $p(x) = x^2 + \alpha^5 x + 1$ over $\mathbb{F}_{2^4}$, where $\alpha^4 + \alpha + 1 = 0$. (hint on course home page, FAQ section)

Laboratory exercise 1: For each of the polynomials below, determine whether the polynomial is primitive, irreducible, or reducible over the specified field.

1. $p(x) = x^{23} + x^5 + 1$ over $\mathbb{F}_2$.
2. $p(x) = x^{23} + x^6 + 1$ over $\mathbb{F}_2$.
3. $p(x) = x^{18} + x^3 + 1$ over $\mathbb{F}_2$.
4. $p(x) = x^8 + x^6 + 1$ over $\mathbb{F}_7$.
5. $p(x) = x^6 + \alpha^5 x + 1$ over $\mathbb{F}_{2^4}$, where $\alpha^4 + \alpha + 1 = 0$. (voluntary)

Home exercise 2: Let $\mathbb{F}_{2^4}$ be constructed through $\pi(x) = x^4 + x + 1$ and $\pi(\alpha) = 0$ ($\alpha^4 + \alpha + 1 = 0$). Determine the *order* of the following elements in $\mathbb{F}_{2^4}$.

1. α.
2. α^2.
3. α^3.
4. α^5.

Laboratory exercise 2: Let $\mathbb{F}_{2^{18}}$ be constructed through $\pi(x) = x^{18}+x^3+1$ and $\pi(\alpha) = 0$ ($\alpha^{18}+\alpha^3+1 = 0$). Determine the *order* of the following elements in $\mathbb{F}_{2^{18}}$.

1. α.
2. α^2.
3. α^3.
4. $\alpha + \alpha^3$.

Finally, we connect the properties of the polynomials and the cycle sets they generate.

Home exercise 3: Determine the cycle set for the first two poynomials in HE 1.

Laboratory exercise 3: Determine the cycle set for the first two poynomials in LE 1.

To help you with the next part of Project 2, we include the following two exercises.

Home exercise 4: Find a primitive polynomial of degree 4 over $\mathbb{F}_2$. In order to facilitate your job in HE 5, choose such a polynomial with constant term 1 (one).

Laboratory exercise 4: Find a primitive polynomial of degree 4 over $\mathbb{F}_5$. Same here, in order to facilitate your job in LE 5, choose such a polynomial with constant term 1 (one).

3 De Bruijn Sequences

A de Bruijn sequence is a sequence with period q^L generated by a feedback shift register (FSR) of length L. A de Bruijn sequence has the maximal possible period for a shift register of length L.

If we consider the cycle set of a de Bruijn sequence, it will contain only one single cycle, running through all q^L different states. It is quite obvious that such a sequence cannot be generated by a linear feedback shift register. We must use *nonlinear* feedback.

One way to proceed is to start with a primitive feedback polynomial. Such a polynomial has two cycles, one containing the all zero state $\mathbf{0}$ only, and another containing the remaining $q^L - 1$ different states. Assume that we have a sequence of transitions $\ldots \mathbf{s} \to \mathbf{s}' \to \ldots$ in the big cycle. The idea is to put the zero state $\mathbf{0}$ somewhere in the big cycle by adding a nonlinear part to the linear feedback. This should result in a change of the original big cycle to instead go through the transitions $\ldots \mathbf{s} \to \mathbf{0} \to \mathbf{s}' \to \ldots$.

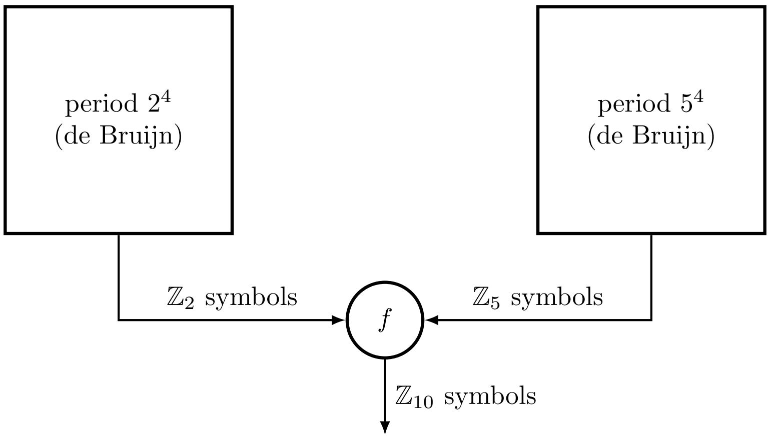
Home exercise 5: Draw a picture of a device that generates a de Bruijn sequence of length 16, using the above described method and the feedback polynomial you derived in HE 4. Use for example XOR and AND gates for the feedback,

Laboratory exercise 5: It is common in different buildings to require a pin code to open certain doors. Usually, the pin code consists of four digits between 0 and 9. So there are 10,000 different pin codes possible. If you do not know better, you might think that it is necessary to push a maximum of 40,000 digits, or on average 20,000 digits, in order to find the correct pin code and open the door.

But this is of course wrong, which is easily seen if you push the sequence "00001" of length 5. You test both pin codes "0000" and "0001". We see that the best we can do is to test a new pin code every time we push a new digit (except the first three). This is exactly a de Bruijn sequence over $\mathbb{Z}_{10}$. This means that we need to push at most 10,003 digits and on average around 5,000.

Your task in this exercise is to write a program that *efficiently* constructs such a de Bruijn sequence over $\mathbb{Z}_{10}$. Write the 10,003 digits to a file and then verify that the sequence runs through every state (every 4-tuple) in $\mathbb{Z}_{10}^4$ exactly once! Check the web page of the course for such a verification program.

Hint 1: As $10 = 5 \cdot 2$ is not a prime, $\mathbb{Z}_{10}$ is not a field, and we cannot construct a maximal length sequence directly. Instead, construct two de Bruijn sequences, one of period 2^4 and one of period 5^4, and then combine them to a single sequence over $\mathbb{Z}_{10}$ by a one-to-one mapping $f : \mathbb{Z}_2 \times \mathbb{Z}_5 \to \mathbb{Z}_{10}$, as illustrated below.



You can think of the two devices as black boxes that output one symbol (each) per clocking. Placing the outputs "side by side" and regarding them as pairs, you need to define the mapping f so that it maps any such pair of symbols uniquely (a bijection) to a symbol in $\mathbb{Z}_{10}$.

You have already constructed the leftmost de Bruijn device in HE 5. Use the same approach to construct the rightmost one.

Hint 2: To efficiently construct the sequence, you cannot rely on a recursive algorithm. Instead, follow the advice given in Hint 1.