# MT5462 ADVANCED CIPHER SYSTEMS

MARK WILDON

These notes cover the part of the syllabus for MT5462 that is not part of the undergraduate course. Further installments will be issued as they are ready. All handouts and problem sheets will be put on the MT362 Moodle page, marked **M.Sc.**

I would very much appreciate being told of any corrections or possible improvements to these notes.

You are warmly encouraged to ask questions in lectures, and to talk to me after lectures and in my office hours. I am also happy to answer questions about the lectures or problem sheets by email. My email address is `mark.wildon@rhul.ac.uk`.

**Lectures:** Monday 4pm (MFLEC), Friday 11am (MC201), Friday 4pm (MC336).

**Extra lecture for MSc students:** Friday 9am (MC201).

**Office hours in McCrea 240:** Tuesday 3.30pm, Wednesday 10am, Thursday 11am or by appointment.

*Date*: First time 2017/18.

## Overview

We start with a secret sharing scheme related to Reed–Solomon codes. We then look at boolean functions, the Berlekamp–Massey algorithm and the Discrete Fourier Transform, and see how these mathematical ideas have been applied to stream ciphers and block ciphers.

## 1. Revision of fields and polynomials

Essentially every modern cipher makes use of the finite field $\mathbb{F}_2$. Many use other finite fields: for example, a fundamental building block in AES (Advanced Encryption Standard) is the inversion map $x \mapsto x^{-1}$ on the finite field $\mathbb{F}_{2^8}$ with 256 elements.

This section should give enough background for the course. It will also be useful for MT5461 Theory of Error Correcting Codes, next term. Proofs in this section are non-examinable.

*Fields.* Informally, a field is a set in which one can add, subtract and multiply any two elements, and also divide by non-zero elements. Examples of infinite fields are the rational numbers $\mathbb{Q}$ and the real numbers $\mathbb{R}$. If $p$ is a prime, then the set $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$, with addition and multiplication defined modulo $p$ is a finite field: see Theorem 1.3.

The formal definition is below. You do not need to memorise this.

**Definition 1.1.** A *field* is a set of elements $\mathbb{F}$ with two operations, $+$ (addition) and $\times$ (multiplication), and two special elements $0,\ 1 \in \mathbb{F}$ such that $0 \neq 1$ and

(1) $a + b = b + a$ for all $a, b \in \mathbb{F}$;
(2) $0 + a = a + 0 = a$ for all $a \in \mathbb{F}$;
(3) for all $a \in \mathbb{F}$ there exists $b \in \mathbb{F}$ such that $a + b = 0$;
(4) $a + (b + c) = (a + b) + c$ for all $a, b, c \in \mathbb{F}$;

(5) $a \times b = b \times a$ for all $a, b \in \mathbb{F}$;
(6) $1 \times a = a \times 1 = a$ for all $a \in \mathbb{F}$;
(7) for all non-zero $a \in \mathbb{F}$ there exists $b \in \mathbb{F}$ such that $a \times b = 1$;
(8) $a \times (b \times c) = (a \times b) \times c$ for all $a, b, c \in \mathbb{F}$;

(9) $a \times (b + c) = a \times b + a \times c$ for all $a, b, c \in \mathbb{F}$.

If $\mathbb{F}$ is finite, then we define its *order* to be its number of elements.

It may be helpful to note that (1)–(4) imply that $\mathbb{F}$ is an abelian group under addition, and that (5)–(8) imply that $(\mathbb{F}\backslash\{0\}, \times)$ is an abelian group under multiplication. The final axiom (9) is the *distributive law* relating addition and multiplication.

It is usual to write $-a$ for the element $b$ in (4); we call $-a$ the *additive inverse* of $a$. We write $a^{-1}$ for the element $b$ in (8); we call $a^{-1}$ the *multiplicative inverse* of $a$. We usually write $ab$ rather than $a \times b$.

*Exercise:* Show, from the field axioms, that if $x \in \mathbb{F}$, then $x$ has a unique additive inverse, and that if $x \neq 0$ then $x$ has a unique multiplicative inverse. Show also that if $\mathbb{F}$ is a field then $a \times 0 = 0$ for all $a \in \mathbb{F}$.

*Exercise:* Show from the field axioms that if $\mathbb{F}$ is a field and $a, b \in \mathbb{F}$ are such that $ab = 0$, then either $a = 0$ or $b = 0$.

We will use the second exercise above many times.

**Theorem 1.2.** *Let $p$ be a prime. The set $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$ with addition and multiplication defined modulo $p$ is a finite field of order $p$.*

There is a unique (up to a suitable notion of isomorphism) finite field of any given prime-power order. The smallest field not of prime order is the finite field of order 4.

**Example 1.3.** The addition and multiplication tables for the finite field $\mathbb{F}_4 = \{0, 1, \alpha, 1+\alpha\}$ of order 4 are shown below.

| $+$ | $0$ | $1$ | $\alpha$ | $1+\alpha$ |
|---|---|---|---|---|
| $0$ | $0$ | $1$ | $\alpha$ | $1+\alpha$ |
| $1$ | $1$ | $0$ | $1+\alpha$ | $\alpha$ |
| $\alpha$ | $\alpha$ | $1+\alpha$ | $0$ | $1$ |
| $1+\alpha$ | $1+\alpha$ | $\alpha$ | $1$ | $0$ |

| $\times$ | $1$ | $\alpha$ | $1+\alpha$ |
|---|---|---|---|
| $1$ | $1$ | $\alpha$ | $1+\alpha$ |
| $\alpha$ | $\alpha$ | $1+\alpha$ | $1$ |
| $1+\alpha$ | $1+\alpha$ | $1$ | $\alpha$ |

Probably the most important thing to realise is that $\mathbb{F}_4$ **is not the integers modulo** 4. Indeed, in $\mathbb{Z}_4 = \{0, 1, 2, 3\}$ we have $2 \times 2 = 0$, but if $a \in \mathbb{F}_4$ and $a \neq 0$ then $a \times a \neq 0$, as can be seen from the multiplication table. (Alternatively this follows from the second exercise above.)

*Polynomials.* Let $\mathbb{F}$ be a field. Let $\mathbb{F}[x]$ denote the set of all polynomials

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_mx^m$$

where $m \in \mathbb{N}_0$ and $a_0, a_1, a_2, \ldots, a_m \in \mathbb{F}$.

**Definition 1.4.** If $f(x) = a_0 + a_1x + a_2 + \cdots + a_mx^m$ where $a_m \neq 0$, then we say that $m$ is the *degree* of the polynomial $f$, and write $\deg f = m$. We leave the degree of the zero polynomial undefined. We say that $a_0$ is the *constant term*.

It is often useful that the constant term in a polynomial $f$ is $f(0)$.

A polynomial is a non-zero constant if and only if it has degree 0. The degree of the zero polynomial is not entirely standardized: you might also see it defined to be $-\infty$, or as $-1$. For this reason we will phrase some results, such as Lemma 1.5 below, so that the whole issue is avoided.

Polynomials are added and multiplied in the natural way.

**Lemma 1.5** (Division algorithm)**.** *Let $\mathbb{F}$ be a field, let $g(x) \in \mathbb{F}[x]$ be a non-zero polynomial and let $f(x) \in \mathbb{F}[x]$. There exist polynomials $s(x), r(x) \in \mathbb{F}[x]$ such that*

$$f(x) = s(x)g(x) + r(x)$$

*and either $r(x) = 0$ or $\deg r(x) < \deg g(x)$.*

We say that $s(x)$ is the *quotient* and $r(x)$ is the *remainder* when $f(x)$ is divided by $g(x)$. Lemma 1.5 will not be proved in lectures. The important thing is that you can find the quotient and remainder in practice. In MATHEMATICA use `PolynomialQuotientRemainder`.

**Exercise 1.6.** Let $g(x) = x^3 + x + 1 \in \mathbb{F}_2[x]$, let $f(x) = x^5 + x^2 + x \in \mathbb{F}_2[x]$. Find the quotient and remainder when $f(x)$ is divided by $g(x)$.

For Shamir's secret sharing scheme we shall need the following properties of polynomials.

**Lemma 1.7.** *Let $\mathbb{F}$ be a field.*

*(i) If $f \in \mathbb{F}[x]$ has $a \in \mathbb{F}$ as a root, i.e. $f(a) = 0$, then there is a polynomial $g \in \mathbb{F}[x]$ such that $f(x) = (x - a)g(x)$.*

*(ii) If $f \in \mathbb{F}[x]$ has degree $m \in \mathbb{N}_0$ then $f$ has at most $m$ distinct roots in $\mathbb{F}$.*

*(iii) Suppose that $f$, $g \in \mathbb{F}[x]$ are non-zero polynomials such that $\deg f$, $\deg g < t$. If there exist distinct $c_1, \ldots, c_t \in \mathbb{F}$ such that $f(c_i) = g(c_i)$ for each $i \in \{1, \ldots, t\}$ then $f = g$.*

Part (iii) is the critical result. It says, for instance, that a linear polynomial is determined by any two of its values: when $\mathbb{F}$ is the real numbers $\mathbb{R}$ this should be intuitive—there is a unique line through any two distinct points. Similarly a quadratic is determined by any three of its values, and so on.

We also need a result on polynomial interpolation that has a surprisingly quick direct proof.

**Lemma 1.8** (Polynomial interpolation)**.** *Let $\mathbb{F}$ be a field. Let*

$$c_1, c_2, \ldots, c_t \in \mathbb{F}$$

*be distinct and let $y_1, y_2, \ldots, y_t \in \mathbb{F}$. The unique polynomial $f(x) \in \mathbb{F}[x]$, either zero or of degree $< t$, such that $f(c_i) = y_i$ for all $i$ is*

$$f(x) = \sum_{i=1}^{t} y_i \frac{\prod_{j \neq i}(x - c_j)}{\prod_{j \neq i}(c_i - c_j)}.$$

## 2. Shamir's secret sharing scheme

*Motivation.* Some flavour of secret sharing is given by the following informal example.

**Example 2.1.** Ten people want to know their mean salary. But none is willing to reveal her salary $s_i$ to the others, or to a 'Trusted Third Party'. Instead Person 1 chooses a large number $M$. She remembers $M$, and whispers $M + s_1$ to Person 2. Then Person 2 whispers $M + s_1 + s_2$ to Person 3, and so on, until finally Person 10 whispers $M + s_1 + s_2 + \cdots + s_{10}$ to Person 1. Person 1 then subtracts $N$ and can tell everyone the mean $(s_1 + s_2 + \cdots + s_{10})/10$.

**Exercise 2.2.** Show that if Person $j$ hears $N$ from Person $j - 1$ then $s_1 + \cdots + s_{j-1}$ can consistently be any number between 0 and $N$.

Provided $M$ is chosen much larger than any conceivable salary, this exercise shows that the scheme does not leak any unintended information.

**Exercise 2.3.** In the two person version of the scheme, Person 1 can deduce Person 2's salary from $N + s_1 + s_2$ by subtracting $N + s_1$. Is this a defect in the scheme?

*Shamir's secret sharing scheme.* In Shamir's scheme the *secret* is an element of a finite field $\mathbb{F}_p$. It will be shared across $n$ people so that any $t$ of them, working together, can deduce the secret, but any $t-1$ of them can learn nothing. To set up the scheme requires a Trusted Third Party, who we will call Trevor.

In a typical application, you are Trevor, and the $n$ people are $n$ untrusted cloud computers. The people are labelled from 1 up to $n$; everyone knows who has which label.

**Definition 2.4.** Let $p$ be a prime and let $s \in \mathbb{F}_p$. Let $n \in \mathbb{N}$, $t \in \mathbb{N}$ be such that $t \leq n < p$. Let $c_1, \ldots, c_n \in \mathbb{F}_p$ be distinct non-zero elements. In the *Shamir scheme* with $n$ people and *threshold* $t$, Trevor chooses at random $a_1, \ldots, a_{t-1} \in \mathbb{F}_p$ and constructs the polynomial

$$f(x) = s + a_1 x + \cdots + a_{t-1} x^{t-1}$$

with constant term $s$. Trevor then issues the *share* $f(c_i)$ to Person $i$.

As often the case in cryptography and coding theory, it is important to be clear about what is private and what is public information.

In the Shamir scheme the parameters $n$, $t$ and $p$ are public, as are the evaluation points $c_1, \ldots, c_n$. Only Trevor knows $f(x)$, and, at the time it is issued, the share $f(c_i)$ is known only to Person $i$ and Trevor.

**Example 2.5.** Suppose that $n = 5$ and $t = 3$. Take $p = 7$ and $c_i = i$ for each $i \in \{1, 2, 3, 4, 5\}$. We suppose that $s = 5$. Trevor chooses $a_1, a_2 \in \mathbb{F}_7$ at random, getting $a_1 = 4$ and $a_2 = 1$. Therefore $f(x) = 5 + 6x + x^2$ and the share of Person $i$ is $f(c_i)$, for each $i \in \{1, 2, 3, 4, 5\}$, so

$$\big(f(1), f(2), f(3), f(4), f(5)\big) = (5, 0, 4, 3, 4).$$

The following exercise shows the main idea needed to prove Theorem 2.7 below.

**Exercise 2.6.** Suppose that Person 1, with share $f(1) = 5$, and Person 2, with share $f(2) = 0$, cooperate in an attempt to discover $s$. Show that for each $z \in \mathbb{F}_7$ there exists a unique polynomial $f_z(x)$ such that $\deg f \leq 2$ and $f(0) = z$, $f_z(1) = 5$ and $f_z(2) = 0$. For example $f_2(x) = 3x^2 + 2$ and $f_3(x) = 2x + 3$. Since Trevor chose the coefficients of $f$ at random, Persons 1 and 2 learn nothing about $s$.

**Theorem 2.7.** *In a Shamir scheme with $n$ people, threshold $t$ and secret $s$, any $t$ people can determine $s$ but any $t-1$ people can learn nothing about $s$.*

The proof shows that any $t$ people can determine the polynomial $f$. So as well as learning $s$, they can also learn the shares of all the other participants.

The remainder of this section is non-examinable and included for interest only.

**Example 2.8.** The root key for DNSSEC, part of web of trust that guarantees an IP connection really is to the claimed end-point, and not Malcolm doing a Man-in-the-Middle attack, is protected by a secret sharing scheme with $n = 7$ and $t = 5$: search for 'Schneier DNSSEC'.

The search above will take you to Bruce Schneier's blog. It is highly recommended for background on practical cryptography.

**Exercise 2.9.** Take the Shamir scheme with threshold $t$ and evaluation points $1, \ldots, n \in \mathbb{F}_p$ where $p > n$. Trevor has shared two large numbers $r$ and $s$ across $n$ cloud computers, using polynomials $f$ and $g$ so that the shares are $\big(f(1), \ldots, f(n)\big)$ and $\big(g(1), \ldots, g(n)\big)$.

(a) How can Trevor secret share $r + s \bmod p$?

(b) How can Trevor secret share $rs \bmod p$?

Note that all the computation has to be done on the cloud!

**Exercise 2.10.** Suppose Trevor shares $s \in \mathbb{F}_p$ across $n$ computers using the Shamir scheme with threshold $t$. He chooses $t$ computers and gets them to reconstruct $s$. Unfortunately Malcolm has compromised one of these computers. Show that Malcolm can both learn $s$ and trick Trevor into thinking his secret is any chosen $s' \in \mathbb{F}_p$.

**Remark 2.11.** The *Reed–Solomon code* associated to the parameters $p$, $n$, $t$ and the field elements $c_1, c_2, \ldots, c_n$ is the length $n$ code over $\mathbb{F}_p$ with codewords all possible $n$-tuples

$$\{\big(f(c_1), f(c_2), \ldots, f(c_n)\big) : f \in \mathbb{F}_p[x],\ \deg f \leq t-1\}.$$

It will be studied in MT5461. By Theorem 2.7, each codeword is determined by any $t$ of its positions. Thus two codewords agreeing in $n - t + 1$ positions are equal: this shows the Reed–Solomon code has minimum distance at least $n - t + 1$.

For simplicity we have worked over a finite field of prime order in this section. Reed–Solomon codes and the Shamir secret sharing scheme generalize in the obvious way to arbitrary finite fields. For example, the Reed–Solomon codes used on compact discs have alphabet the finite field $\mathbb{F}_{2^8}$.

## 3. Introduction to Boolean functions

The finite field $\mathbb{F}_2$ is the set of *bits* $\{0,1\}$, with addition and multiplication defined modulo 2.

**Definition 3.1.** Let $n \in \mathbb{N}$. An $n$-variable *boolean function* is a function $\mathbb{F}_2^n \to \mathbb{F}_2$.

Most modern ciphers are defined by combining boolean functions in various ways.

We think of $0 \in \mathbb{F}_2$ as *false* and $1 \in \mathbb{F}_2$ as *true*. For example, with this convention, logical and is the Boolean function $f(x,y) = xy$, since $xy = 1$ if and only if $x = 1$ and $y = 1$.

Any boolean function is uniquely determined by its *truth table*, which records the pairs $\big(x, f(x)\big)$ for each $x \in \mathbb{F}_2^n$. For example, the truth tables for and, or (denoted $\vee$) and not (denoted $\neg$) are shown below.

| $x$ | $y$ | $xy$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| $x$ | $y$ | $x \vee y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| $x$ | $\neg x$ |
|---|---|
| 0 | 1 |
| 0 | 0 |

**Exercise 3.2.** Show that there are $2^{2^n}$ boolean function in $n$ variables.

**Exercise 3.3.** Which boolean functions have the truth tables below?

| $x$ | $y$ | |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| $x$ | $y$ | |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Boolean functions can be expressed in many different ways, not always obviously the same. In this section we look at some normal forms for Boolean functions.

The following examples will be useful to bear in mind.

**Exercise 3.4.**

(i) Write the two variable function $f(x,y) = x \vee y$ as a polynomial in $x$ and $y$.

(ii) What logical connective corresponds to $(x,y) \mapsto x + y$?

(iii) Define $\mathrm{maj}(x_1, x_2, x_3)$ to be true if at least two of $x_1, x_2, x_3$ are true, and otherwise false. Express maj as a polynomial.

(iv) Express $x_1x_2 \vee x_2x_3 \vee x_3x_4$ as a polynomial.

*Algebraic normal form.* In $\mathbb{F}_2$ we have $0^2 = 0$ and $1^2 = 1$. Therefore the Boolean functions $f(x) = x^2$ and $f(x) = x$ are equal. This means we do not need squares or any higher powers of the variables. Similarly, since $2x_1 = 0$, the only coefficients we need are the bits 0 and 1.

For instance, $(x_1 + x_1x_2^2x_3^3) + (x_1 + x_2)$ is the same function as $x_2 + x_1x_2x_3$.

**Exercise 3.5.** Find a simple form for the product of $f(x_1, x_2, x_3) = x_1(\neg x_2)x_3$ and $\mathrm{maj}(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3x_1$.

We define a *boolean monomial* to be a product of the form $x_{i_1} \dots x_{i_r}$ where $i_1 < \dots < i_r$. Given $I \subseteq \{1, \dots, n\}$, let

$$x_I = \prod_{i \in I} x_i.$$

By definition (or convention if you prefer), $x_\varnothing = 1$.

For example, $x_{\{2,3\}} = x_2x_3$. It is one of the three monomial summands of $\mathrm{maj}(x_1, x_2, x_3)$.

**Lemma 3.6.** *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a Boolean function. Then*

$$f(x_1, x_2, \dots, x_n) = x_1 g(x_2, \dots, x_n) + (1 + x_1)h(x_2, \dots, x_n)$$

*where $g(x_2, \dots, x_n) = f(1, x_2, \dots, x_n)$ and $h(x_2, \dots, x_n) = f(0, x_2, \dots, x_n)$.*

Particularly if you have done some programming, it may be helpful to think of $x_1g + (1 + x_1)h$ as 'if $x_1$ is true, then $g$, else $x_1$ is false, and so $h$'.

By repeatedly applying Lemma 3.6 one can write an arbitrary boolean function in $n$ variables as a polynomial in $x_1, \dots, x_n$, and so as a sum of boolean monomials.

**Example 3.7.** The Toffoli gate is important in quantum computation. It takes 3 input qubits and returns 3 output qubits. Its classical analogue is the 3 variable Boolean function defined in words by 'if $x_1$ and $x_2$ are both true then negate $x_3$, else return $x_3$'. Using Lemma 3.6, one gets the polynomial form $x_1x_2 + x_3$.

**Theorem 3.8.** *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be an $n$-variable Boolean function. There exist unique coefficients $c_I \in \{0, 1\}$, one for each $I \subseteq \{1, \dots, n\}$, such that*

$$f(x_1, \dots, x_n) = \sum_{I \subseteq \{1, \dots, n\}} c_I x_I.$$

This expression for $f$ is called the *algebraic normal form* of $f$.

It is possible to give an explicit formula for the coefficients $c_I$ in the algebraic normal form. It can be guessed by looking at some small examples.

**Example 3.9.** Let $f : \mathbb{F}_2^3 \to \mathbb{F}_2$ be a 3-variable Boolean function

(a) Show that the coefficient $c_\varnothing$ of $x_\varnothing = 1$ in $f$ is $f(0,0,0)$.
(b) Show that the coefficient $c_{\{3\}}$ of $x_{\{3\}} = x_3$ in $f$ is $f(0,0,0) + f(0,0,1)$.
(c) Show that the coefficient $c_{\{1,2\}}$ of $x_{\{1,2\}} = x_1x_2$ in $f$ is $f(0,0,0) + f(1,0,0) + f(0,1,0) + f(1,1,0)$.

For example, by (c), if $f(x_1,x_2,x_3) = x_1x_2 + x_3$ is the Toffoli function seen in Example 3.7 then $f(0,0,0) + f(1,0,0) + f(0,1,0) + f(1,1,0) = 0+0+0+1 = 1$ is the coefficient of $x_1x_2$.

**Exercise 3.10.** What do you think is the formula for the coefficient $c_{\{2,3\}}$? Does it work for the Toffoli function? How about if $f(x_1,x_2,x_3) = x_1x_2x_3$?

**Proposition 3.11.** *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be an $n$-variable Boolean function and suppose that $f$ has algebraic normal form*

$$f(x_1,\dots,x_n) = \sum_{I \subseteq \{1,\dots,n\}} c_I x_I.$$

*Then*

$$c_I = \sum f(z_1,\dots,z_n)$$

*where the sum is over all $z_1,\dots,z_n \in \{0,1\}$ such that $\{j : z_j = 1\} \subseteq I$.*

*Disjunctive normal form.* For the remaining normal forms it is best to think of $0 \in \mathbb{F}_2$ as false and $1 \in \mathbb{F}_2$ as true. Following the usual convention, we write $\wedge$ for 'and' (also called *conjunction*). Thus $x \wedge y = xy$ is the conjunction of variables $x$ and $y$. Recall that $\vee$ denotes 'or' (also called *disjunction*) and $\neg$ negation.

**Definition 3.12.** Fix $n \in \mathbb{N}$. Given $J \subseteq \{1,\dots,n\}$ let

$$f_J(x_1,\dots,x_n) = z_1 \wedge \cdots \wedge z_n$$

where

$$z_j = \begin{cases} x_j & \text{if } j \in J \\ \neg x_j & \text{if } j \notin J. \end{cases}$$

A $n$-variable Boolean function of the form $\bigvee_{J \in \mathcal{B}} f_J$, where $\mathcal{B}$ is a collection of subsets of $\{1,\dots,n\}$, is said to be in *disjunctive normal form*.

By definition, or convention if you prefer, the empty disjunction is false; thus $f_\varnothing = 0$.

For example $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2) \vee (x_1 \wedge x_2)$. is in disjunctive normal form. The collection $\mathcal{B}$ in the definition is $\{\{1\},\{2\},\{1,2\}\}$. What is this function in words?

**Example 3.13.**

(a) We saw in Exercise 3.4 that $\operatorname{maj}(x_1,x_2,x_3) = (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_1 \wedge x_3)$. From this it is a short step to the disjunctive normal form

$$\begin{aligned}\operatorname{maj}(x_1,x_2,x_3) &= (x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \\ &\qquad \vee (\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) \\ &= f_{\{1,2\}} \vee f_{\{1,3\}} \vee f_{\{2,3\}} \vee f_{\{1,2,3\}}\end{aligned}$$

(b) The truth table for the Toffoli function $f(x_1,x_2,x_3) = x_1x_2 + x_3$ is

| $x_1$ | $x_2$ | $x_3$ | $f$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |

| $x_1$ | $x_2$ | $x_3$ | $f$ |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

So $f(x_1,x_2,x_3)$ is true if and only the set of true variables is one of $\{3\}$, $\{2,3\}$, $\{1,3\}$ or $\{1,2\}$. Correspondingly,

$$\begin{aligned}f(x_1,x_2,x_3) &= (\neg x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3) \\ &\qquad \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3). \\ &= f_{\{3\}} \vee f_{\{2,3\}} \vee f_{\{1,3\}} \vee f_{\{1,2\}}.\end{aligned}$$

**Theorem 3.14.** *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be an $n$-variable Boolean function. There exists a unique collection $\mathcal{B}$ of subsets of $\{1,\ldots,n\}$ such that*

$$f(x_1,\ldots,x_n) = \bigvee_{J\in\mathcal{B}} f_J.$$

*Conjunctive normal form.* Given a Boolean formula $f$ expressed using $\vee$ and $\wedge$, one obtains $\neg f$ by swapping $\vee$ and $\wedge$ and negating every variable. For example, $x_1 \vee x_2$ becomes $\neg x_1 \wedge \neg x_2$ which equals $\neg(x_1 \vee x_2)$.

Conjunctive normal form is obtained from disjunctive normal form by this duality.

**Definition 3.15.** Fix $n \in \mathbb{N}$. Given $J \subseteq \{1,\ldots,n\}$, let $g_J = z_1 \vee \cdots \vee z_n$ where, as in Definition 3.12,

$$z_j = \begin{cases} x_j & \text{if } j \in J \\ \neg x_j & \text{if } j \notin J. \end{cases}.$$

A Boolean function of the form $\bigvee_{J\in\mathcal{B}} g_J$, where $\mathcal{B}$ is a collection of subsets of $\{1,\ldots,n\}$, is said to be in *conjunctive normal form*.

Given $f : \mathbb{F}_2^n \to \mathbb{F}_2$ one can write $f$ in conjunctive normal form by writing $\neg f$ in disjunctive normal form and then negating it, using that if $J \subseteq \{1,\ldots,n\}$ then $\neg f_J = g_{J'}$ where $J' = \{k \in \{1,\ldots,n\} : k \not\in J\}$.

**Example 3.16.** The majority vote function maj on 3-variables is false if and only if at least two of the variables are false. Hence $\neg\text{maj}(x_1,x_2,x_3) = f_\varnothing \vee f_{\{1\}} \vee f_{\{2\}} \vee f_{\{3\}}$ in disjunctive normal form and so

$$\begin{aligned}
\text{maj}(x_1,x_2,x_3) &= \neg\big(f_\varnothing \vee f_{\{1\}} f_{\{2\}} \neg f_{\{3\}}\big)\\
&= \neg f_\varnothing \wedge \neg f_{\{1\}} \wedge \neg f_{\{2\}} \wedge \neg f_{\{3\}}\\
&= g_{\{1,2,3\}} \wedge g_{\{2,3\}} \wedge g_{\{1,3\}} \wedge g_{\{1,2\}}\\
&= (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)\\
&\qquad\qquad \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)
\end{aligned}$$

in conjunctive normal form.

## 4. Berlekamp–Massey Algorithm

The Berlekamp–Massey Algorithm finds an LFSR of minimal width generating a given keystream. It is a faster algorithm than the linear algebra method seen in Question 1 of Sheet 5.

Such an LFSR always exists, since given $(k_0,k_1,\ldots,k_{n-1}) \in \mathbb{F}_2^\ell$ we can simply take any LFSR of width $n$ and the entire keystream as the key. But there may be an LFSR of smaller width that works.

*Preliminaries.* If $F$ is an LFSR width $\ell$ with taps $T \subseteq \{0,1,\ldots,\ell-1\}$ then, for each position $s \in \mathbb{N}$,

$$F(k_s,\ldots,k_{s+\ell-1}) = (k_{s+1},\ldots,k_{s+\ell-1},\sum_{t\in T} k_{s+t}).$$

Hence (as seen in Question 1 of Sheet 5), $k_{s+\ell} = \sum_{t\in T} k_{s+t}$. Setting $r = s+\ell$ this becomes

$$(\dagger) \qquad k_r = \sum_{t\in T} k_{r-\ell+t} \quad \text{for } r \geq \ell.$$

Recall from page 8 that $\neg 0 = 1$ and $\neg 1 = 0$. (Equivalently, $\neg c = c+1$.)

**Proposition 4.1.** *Let $n \geq \ell$. If an LFSR $F$ of width $\ell$ generates the keystream $(k_0,k_1,\ldots,k_{n-1},c)$ of length $n+1$ then any LFSR $F'$ generating the keystream $(k_0,k_1,\ldots,k_{n-1},\neg c)$ has width $\ell'$ where $\ell' \geq n+1-\ell$.*

*Proof.* Suppose, for a contradiction that $\ell' \le n - \ell$. Let $T$ be the set of taps of $F$ and let $T'$ be the set of taps of $F'$. By (†) for $F'$ we have

$$(\dagger') \qquad k_r = \sum_{t' \in T'} k_{r-\ell'+t'} \quad \text{for } r \ge \ell'.$$

By (†) in the case $r = n$ we have

$$c = \sum_{t \in T} k_{n-\ell+t}.$$

Since $n - \ell \ge \ell'$, the equation (†′) holds when $r \ge n - \ell$. Substituting for each $k_{n-\ell+t}$ using this equation we get

$$c = \sum_{t \in T} \sum_{t' \in T} k_{n-\ell+t-\ell'+t'} = \sum_{t' \in T} \sum_{t \in T} k_{n+t'-\ell'-\ell+t} = \sum_{t' \in T} k_{n+t'-\ell'} = \neg c$$

where we swapped the order of summation, then used (†), then (†′). Hence $c = \neg c$, a contradiction. □

As a final preliminary, we need the *symmetric difference* of sets $T$ and $U$ defined by

$$T \triangle U = \{v \in T \cup U : v \notin T \cap U\}.$$

The following lemma shows how symmetric differences arise when we combine LFSRs.

**Lemma 4.2 (corrected).** *Let $F$ and $G$ be LFSRs of width $\ell$ with taps $T$ and $U$ respectively. The function $H$ defined by*

$$H\big((x_0, \ldots, x_{\ell-1})\big) = (x_1, \ldots, x_{\ell-1}, \sum_{t \in T} x_t + \sum_{u \in U} x_u)$$

*is an LFSR with taps $T \triangle U$.*

*Berlekamp–Massey Algorithm.* We define $\widetilde{T} = \{\ell - t : t \in T\}$ so, by (†),

$$(\ddagger) \qquad k_n = \sum_{\widetilde{t} \in \widetilde{T}} k_{n-\widetilde{t}} \qquad \text{for all } n \ge \ell.$$

**Example 4.3.** The keystream of the LFSR $F$ of width 5 with taps $\{0, 1, 2\}$ for the key $(0, 1, 1, 0, 0)$ has period 14.

$$(\underset{0}{0}, \underset{1}{1}, \underset{2}{1}, \underset{3}{0}, \underset{4}{0}, \underset{5}{0}, \underset{6}{0}, \underset{7}{1}, \underset{8}{0}, \underset{9}{0}, \underset{10}{1}, \underset{11}{1}, \underset{12}{1}, \underset{13}{1}, \ldots)$$

The set $\widetilde{T}$ is $\{5-0, 5-1, 5-2\} = \{3, 4, 5\}$ and, as claimed by (‡), $k_n = k_{n-3} + k_{n-4} + k_{n-5}$ for all $n \in \mathbb{N}$ with $n \ge 5$.

We use the following notation in the algorithm.

- $k_0, k_1, k_2, \ldots$ is the keystream;
- for $n \in \mathbb{N}$, $\ell_n$ is the minimal width of an LFSR $F_n$ with taps $T_n$ generating the first $n$ positions $k_0, k_1, \ldots, k_{n-1}$ of the keystream;
- $\widetilde{T}_n = \{\ell_n - t : t \in T_n\}$.

**Theorem 4.4.** *Let $n \in \mathbb{N}$.*

(a) *If the LFSR $F_n$ generates $(k_0, k_1, \dots, k_{n-1}, k_n)$ then $\ell_{n+1} = \ell_n$ and we may take $\widetilde{T}_{n+1} = \widetilde{T}_n$.*

(b) *Suppose the LFSR $F_n$ generates $(k_0, k_1, \dots, k_{n-1}, \neg k_n)$. If $\ell_n = 0$ then let $m = 0$, else let $m$ be maximal such that $\ell_m < \ell_{m+1}$. Let $U = \{\widetilde{t} + n - m : \widetilde{t} \in \widetilde{T}_m\}$. Then setting*

$$\widetilde{T}_{n+1} = \widetilde{T}_n \triangle (U \cup \{n - m\})$$

*defines an LFSR $F_{n+1}$ of minimal width $\ell_{n+1} = \max(\ell_n, n + 1 - \ell_n)$ that generates $(k_0, k_1, \dots, k_{n-1}, k_n)$.*

By definition, in (ii), $\ell_m < \ell_{m+1}$, so at step $m$ we had to increase the width of the LFSR to $\ell_{m+1}$ to get agreement with the first $m + 1$ keystream bits. There could be more recent changes of the taps that kept the width constant at $\ell_{m+1}$.

**Example 4.5.** We take the first 12 positions of the keystream generated by the LFSR in Example 4.3 but change the final 1 to 0.

$$(\underset{0}{0}, \underset{1}{1}, \underset{2}{1}, \underset{3}{0}, \underset{4}{0}, \underset{5}{0}, \underset{6}{0}, \underset{7}{1}, \underset{8}{0}, \underset{9}{0}, \underset{10}{1}, \underset{11}{0}, \dots)$$

The table below shows $\ell_n$ and the set $\widetilde{T}_n$ for each $n$. Where case (ii) applies, the relevant $m$ is shown. The final row indicates whether the set of taps is unique. (This is not given by the algorithm, but can be determined using the linear algebra method.)

| $n$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\ell_n$ | 0 | 2 | 2 | 2 | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 7 |
| $\widetilde{T}_n$ | $\varnothing$ | $\{1\}$ | $\{1\}$ | $\{1,2\}$ | $\{1,2,3\}$ | $\varnothing$ | $\varnothing$ | $\{3,4,5\}$ | $\{3,4,5\}$ | $\{3,4,5\}$ | $\{3,4,5\}$ | $\{3,5\}$ |
| $m$ | 0 | | 1 | 1 | 4 | | 4 | | | | 7 | |
| unique? | ✓ | × | × | ✓ | × | × | ✓ | × | × | ✓ | ✓ | × |

We illustrate the algorithm by taking $n = 1$, $n = 2$ and $n = 5$.

**Init:** Set $\ell_0 = 0$ and $\widetilde{T}_0 = \varnothing$. Define $\ell_1 = 0$ and $\widetilde{T}_1 = \varnothing$. The keystream starts 0, so this defines the unique minimal width LFSR $F_1$.

$n = 1$: the LFSR $F_1$ generates $00\dots$; since $(0,0) = (k_0, \neg k_1)$, case (b) applies. We have $m = 0$ and $U = \{\widetilde{t} + 1 - 0 : \widetilde{t} \in \widetilde{T}_0\} = \{\widetilde{t} + 1 - 0 : \widetilde{t} \in \varnothing\} = \varnothing$. We set

$$\widetilde{T}_2 = \widetilde{T}_1 \triangle (U \cup \{2 - 1\}) = \varnothing \triangle \{1\} = \{1\}$$

and $\ell_2 = \max(\ell_1, 1 + 1 - \ell_1) = \max(0, 2) = 2$.

$n = 2$: the LFSR $F_2$ of width $\ell_2 = 2$ with $\widetilde{T}_2 = \{1\}$ generates $0111\dots$; since $(0,1,1) = (k_0, k_1, k_2)$, case (a) applies and $\ell_3 = \ell_2$, $\widetilde{T}_3 = \widetilde{T}_2$.

$n = 3, 4$: *Exercise:* do these steps.

$n = 5$: the LFSR $F_5$ of width $\ell_5 = 3$ with $\widetilde{T}_5 = \{1, 2, 3\}$ generates 0110011 $\ldots$; since $(0, 1, 1, 0, 0, 1) = (k_0, k_1, k_2, k_3, k_4, \neg k_5)$, case (b) applies. We have $m = 4$ and $U = \{\widetilde{t} + 5 - 4 : \widetilde{t} \in \widetilde{T}_4\} = \{2, 3\}$. We set

$$\widetilde{T}_6 = \widetilde{T}_5 \mathbin{\triangle} (U \cup \{5 - 4\}) = \{1, 2, 3\} \mathbin{\triangle} \{2, 3, 1\} = \varnothing$$

and $\ell_6 = \max(\ell_5, 5 + 1 - \ell_5) = \max(3, 3) = 3$.

$n \geq 7$: *Exercise:* do some more steps, for example $n = 11$.

*Proof of Theorem 4.4.* Set $\ell_0 = 0$ and $T_0 = \varnothing$. We work by induction on $n \in \mathbb{N}$, proving that:

> *Claim:* Theorem 4.4 holds for all $r < n$. Moreover, whenever $r < n$ and $\widetilde{T}_r \neq \widetilde{T}_{r+1}$ or $\ell_{r+1} \neq \ell_r$ equality holds in Proposition 4.1, i.e. $\ell_{r+1} = \max(\ell_r, r + 1 - \ell_r)$.

*Base case:* when $n = 1$ the claim holds vacuously.

*Inductive step:* Suppose the claim holds for $n \in \mathbb{N}$. We must prove Theorem 4.4, as stated above for $n$, and the 'moreover' part of the claim.

For (a), if $F_n$ generates $(k_0, k_1, \ldots, k_{n-1}, k_n)$ then, since no shorter LFSR generates $(k_0, \ldots, k_{n-1})$, $F_n$ is the minimal length LFSR that generates $(k_0, k_1, \ldots, k_{n-1}, k_n)$. The taps and length do not change, so there is nothing more to check.

In (b), we suppose $F_n$ generates $(k_0, k_1, \ldots, k_{n-1}, \neg k_n)$. Then by (‡)

$$\neg k_n = \sum_{\widetilde{t} \in \widetilde{T}_n} k_{n-\widetilde{t}}, \qquad k_s = \sum_{\widetilde{t} \in \widetilde{T}_n} k_{s-\widetilde{t}} \quad \text{if } s < n.$$

Moreover, by choice of $m$, and Theorem 4.4 for $m$, (‡) implies the analogous equations

$$\neg k_m = \sum_{\widetilde{t} \in \widetilde{T}_m} k_{m-\widetilde{t}}, \qquad k_s = \sum_{\widetilde{t} \in \widetilde{T}_m} k_{s-\widetilde{t}} \quad \text{if } s < m.$$

Since $\widetilde{T}_{n+1} = \widetilde{T}_n \mathbin{\triangle} (U \cup \{n - m\})$, where $U = \{\widetilde{t} + n - m : \widetilde{t} \in \widetilde{T}_m\}$, the lemma implies that

$$\begin{aligned}
\sum_{\widetilde{t} \in \widetilde{T}_{n+1}} k_{n-\widetilde{t}} &= \sum_{\widetilde{t} \in \widetilde{T}_n} k_{n-\widetilde{t}} + \sum_{\widetilde{t} \in \widetilde{T}_m} k_{n-(\widetilde{t}+(n-m))} + k_{n-(n-m)} \\
&= \sum_{\widetilde{t} \in \widetilde{T}_n} k_{n-\widetilde{t}} + \sum_{\widetilde{t} \in \widetilde{T}_m} k_{m-\widetilde{t}} + k_m \\
&= \neg k_n + \neg k_m + k_m \\
&= \neg k_n + 1 \\
&= k_n. \ [\textbf{Corrected typo } k_{n+1}]
\end{aligned}$$

Similarly if $s < n$ then,

$$\begin{aligned}\sum_{\widetilde{t}\in\widetilde{T}_{n+1}} k_{s-\widetilde{t}} &= \sum_{\widetilde{t}\in\widetilde{T}_n} k_{s-\widetilde{t}} + \sum_{\widetilde{t}\in\widetilde{T}_m} k_{s-(\widetilde{t}+(n-m))} + k_{s-(n-m)} \\ &= \sum_{\widetilde{t}\in\widetilde{T}_n} k_{s-\widetilde{t}} + \sum_{\widetilde{t}\in\widetilde{T}_m} k_{s-(n-m)-\widetilde{t}} + k_{s-(n-m)} \\ &= k_s + k_{s-n-m} + k_{s-(n-m)} \\ &= k_s.\end{aligned}$$

Hence, by (‡), $F_{n+1}$ generates $(k_0, k_1, \ldots, k_{n-1}, k_n)$.

There are two cases for the width. Since $m$ was the most recent width change, we have, $\ell_{m+1} > \ell_m$ and by induction, $\ell_{m+1} = \max(\ell_m, m+1-\ell_m) = m+1-\ell_m$. Hence

$$\ell_n = m + 1 - \ell_m.$$

(i) If $n - m + \ell_m \le \ell_n$ then $\max U \le \ell_n$ and so $\max \widetilde{T}_{n+1} \le \max \widetilde{T}_n$. Therefore the width need not change. As in (a), no LFSR of width $< \ell_n$ generates $(k_0, k_1, \ldots, k_n, k_{n+1})$. Since we want $F_{n+1}$ to have minimal length we *must* set $\ell_{n+1} = \ell_n$. Now $\ell_{n+1} = m + 1 - \ell_m = m + 1 - \ell_n \le n + 1 - \ell_n$, so $\max(\ell_{n+1}, n+1-\ell_n) = \ell_{n+1}$, as required.

(ii) If $n - m + \ell_m > \ell_n$ then $\max(U) > \ell_n$ and $\max \widetilde{T}_{n+1} = \max U + n - m = \max \widetilde{T}_m + (n-m) \le \ell_m + (n-m)$. Therefore $\ell_{n+1} > \ell_n$ and the least possible choice for $\ell_{n+1}$ is $\ell_{n+1} = (n-m) + \ell_m$. By the displayed equation relating $\ell_n$ and $\ell_m$ above,

$$\ell_{n+1} = (n-m) + (m+1) - \ell_n = n + 1 - \ell_n.$$

Since $\ell_{n+1} > \ell_n$ we have $\max(n+1-\ell_n, \ell_n) = \ell_{n+1}$. Moreover, by Proposition 4.1, no LFSR of width $< n + 1 - \ell_n$ generates $(k_0, k_1, \ldots, k_{n-1}, k_n)$.

This proves the claim for $n+1$, and completes the inductive step. □

The original paper is *Shift-register synthesis and BCH decoding*, James L. Massey, IEEE Transactions on Information Theory, **15** (1969) 122–127. It deals with LFSRs defined over an arbitrary field and leads to an algorithm for decoding cyclic Reed–Solomon codes (and the more general BCH codes in the title). We have simplified things slightly by working over $\mathbb{F}_2$.

The Berlekamp–Massey algorithm is implemented in the code online at `https://repl.it/NE32/4`. Try

```
berlekampMassey [0,1,1,0,0,0,0,1,0,0,1,0]
```

to check Example 4.5; for example, the output line

```
(4,(4,[1,2]),(3,[1,2,3]))
```

shows that when the algorithm is applied with $n = 4$, the new length and taps are $\ell_5 = 3$, $\widetilde{T}_5 = \{1, 2, 3\}$, and $m$ is now 4, with $\widetilde{T}_4 = \{1, 2\}$.

## 5. The Discrete Fourier Transform

In this section it will be useful to change the range of Boolean functions so that they take values in $\{-1, 1\}$ rather than $\{0, 1\}$.

Given $x \in \mathbb{F}_2$ we define $(-1)^x$ by regarding $x$ as an ordinary integer. Thus $(-1)^0 = 1$ and $(-1)^1 = -1$. Given $f : \mathbb{F}_2^n \to \mathbb{F}_2$ we define $(-1)^f : \mathbb{F}_2^n \to \{-1, 1\}$ by $(-1)^f(x) = (-1)^{f(x)}$.

**Definition 5.1.** Let $f, g : \mathbb{F}_2^n \to \mathbb{F}$ be Boolean functions. We define the *correlation* between $f$ and $g$ by

$$\operatorname{corr}(f, g) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} (-1)^{g(x)}.$$

The connection with the correlation statistic used in the main course to compare sequence of bits (see Definition 6.5 and the following results) is shown by the exercise below.

**Lemma 5.2.** *Let $f, g : \mathbb{F}_2^n \to \mathbb{F}$ be Boolean functions. Let*

$$\begin{aligned} c_{\text{same}} &= \left|\{x \in \mathbb{F}_2^n : f(x) = g(x)\}\right| \\ c_{\text{diff}} &= \left|\{x \in \mathbb{F}_2^n : f(x) \neq g(x)\}\right|. \end{aligned}$$

*Then* $\operatorname{corr}(f, g) = (c_{\text{same}} - c_{\text{diff}})/2^n$.

Thus the correlation takes values between 1 (perfect agreement) and $-1$ (always different); as before, 0 can be interpreted as no correlation.

**Exercise 5.3.** Let $X \in \mathbb{F}_2^n$ be a random variable distributed uniformly at random, so $\mathbf{P}[X = x] = 1/2^n$ for each $x \in \mathbb{F}_2^n$. Show that

$$\operatorname{corr}(f, g) = \mathbf{P}[f(X) = g(X)] - \mathbf{P}[f(X) \neq g(X)$$

and

$$\begin{aligned} \mathbf{P}[f(X) = g(X)] &= \tfrac{1}{2}(1 + \operatorname{corr}(f, g)), \\ \mathbf{P}[f(X) \neq g(X)] &= \tfrac{1}{2}(1 - \operatorname{corr}(f, g)). \end{aligned}$$

For example, $\operatorname{corr}(f, g) = 1$ if and only if $f$ and $g$ are the same function, $\operatorname{corr}(f, g) = \frac{1}{2}$ if and only if $\mathbf{P}[f(X) = g(X)] = \frac{3}{4}$ and $\operatorname{corr}(f, g) = 0$ if and only if $\mathbf{P}[f(X) = g(X)] = \mathbf{P}[f(X) \neq g(X)] = \frac{1}{2}$.

We have seen in the main course (see for instance Example 7.1 and 7.2) that linear functions are often weak cryptographically. So are functions that are highly correlated with linear functions.

Given $T \subseteq \{1, \ldots, n\}$, define $L_T : \mathbb{F}_2^n \to \mathbb{F}_2$ by

$$L_T(x) = \sum_{t \in T} x_t.$$

We think of $L_T$ as 'tapping' (like an LFSR) the positions in $T$. For example, $L_{\{i\}}(x_1, \ldots, x_n) = x_i$ returns the entry in position $i$ and $L_\varnothing(x) = 0$ is the zero function.

**Exercise 5.4.** Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. Show that $\mathrm{corr}(f, L_\varnothing) = 0$ if and only if $\mathbf{P}[f(X) = 0] = \mathbf{P}[f(X) = 1] = \frac{1}{2}$.

**Lemma 5.5.** *The linear functions $\mathbb{F}_2^n \to \mathbb{F}$ are precisely the $L_T : \mathbb{F}_2^n \to \mathbb{F}_2$ for $T \subseteq \{1, \ldots, n\}$. If $S, T \subseteq \{1, \ldots, n\}$ then*

$$\mathrm{corr}(L_S, L_T) = \begin{cases} 1 & \textit{if } S = T \\ 0 & \textit{otherwise.} \end{cases}$$

**Example 5.6.** Let $\mathrm{maj} : \mathbb{F}_2^3 \to \mathbb{F}_2$ be the majority vote function from Exercise 3.4(ii). Then

$$\mathrm{corr}(\mathrm{maj}, L_T) = \begin{cases} \frac{1}{2} & \text{if } T = \{1\}, \{2\}, \{3\} \\ -\frac{1}{2} & \text{if } T = \{1, 2, 3\} \\ 0 & \text{otherwise.} \end{cases}$$

Moreover

$$(-1)^{\mathrm{maj}} = \tfrac{1}{2}(-1)^{L_{\{1\}}} + \tfrac{1}{2}(-1)^{L_{\{2\}}} + \tfrac{1}{2}(-1)^{L_{\{3\}}} - \tfrac{1}{2}(-1)^{L_{\{1,2,3\}}}.$$

To generalize the previous example, we define an inner product on the vector space of functions $\mathbb{F}_2^n \to \mathbb{R}$ by

$$\langle \theta, \phi \rangle = \frac{1}{2^n} \sum_{x \in 2^n} \theta(x)\phi(x).$$

*Exercise:* check that, as required for an inner product, $\langle \theta, \theta \rangle \geq 0$ and that $\langle \theta, \theta \rangle = 0$ if and only if $\theta(x) = 0$ for all $x \in \mathbb{F}_2^n$.

**Lemma 5.7.** *Let $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ be Boolean functions. Then*

$$\langle (-1)^f, (-1)^g \rangle = \mathrm{corr}(f, g).$$

**Theorem 5.8** (Discrete Fourier Transform)**.**

(a) *The functions $(-1)^{L_T}$ for $T \subseteq \{1, \ldots, n\}$ are an orthonormal basis for the vector space of functions $\mathbb{F}_2^n \to \mathbb{R}$.*

(b) *Let $\theta : \mathbb{F}_2^n \to \mathbb{R}$. Then*

$$\theta = \sum_{T \subseteq \{1,\ldots,n\}} \langle \theta, (-1)^{L_T} \rangle (-1)^{L_T}.$$

(c) *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. Then*

$$(-1)^f = \sum_{T \subseteq \{1,\ldots,n\}} \mathrm{corr}(f, L_T)(-1)^{L_T}.$$

We call (c) the 'Discrete Fourier Inversion Theorem'. The function $S \mapsto \operatorname{corr}(f, L_S) = \langle (-1)^f, (-1)^{L_S} \rangle$ is the Discrete Fourier Transform of $f$.

## 6. LINEAR CRYPTANALYSIS

In the previous section we considered Boolean functions $\mathbb{F}_2^n \to \mathbb{F}_2$. Typically cryptographic functions return multiple bits, not just one. So we must choose which output bits to tap. In this section we number positions from 0 up to $n-1$.

Recall that $\circ$ denotes composition of functions: thus if $F : \mathbb{F}_2^m \to \mathbb{F}_2^n$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^p$ then $G \circ F : \mathbb{F}_2^m \to \mathbb{F}_2^p$ is the function defined by $(G \circ F)(x) = G\big(F(x)\big)$.

**Example 6.1.** Let $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ be the $S$-box in the $Q$-block cipher (see Example 8.4 in the main notes), defined by

$$S\big((x_0, x_1, x_2, x_3)\big) = (x_2, x_3, x_0 + x_1x_2, x_1 + x_2x_3).$$

(a) Suppose we look at position 0 of the output by considering $L_{\{0\}} \circ S : \mathbb{F}_2^4 \to \mathbb{F}_2$. We have

$$(L_{\{0\}} \circ S)\big((x_0, x_1, x_2, x_3)\big) = x_2 = L_{\{2\}}\big((x_0, x_1, x_2, x_3)\big).$$

Hence $L_{\{0\}} \circ S = L_{\{2\}}$. By Lemma 5.5,

$$\operatorname{corr}(L_{\{0\}} \circ S, L_T) = \begin{cases} 1 & \text{if } T = \{2\} \\ 0 & \text{otherwise.} \end{cases}.$$

(b) Instead if we look at position 2, the relevant Boolean function is $L_{\{2\}} \circ S$, for which $L_{\{2\}} \circ S\big((x_0, x_1, x_2, x_3)\big) = x_0 + x_1x_2$. *Exercise:* show that

$$\operatorname{corr}(L_{\{2\}} \circ S, L_T) = \begin{cases} \frac{1}{2} & \text{if } T = \{0\}, \{0,1\}, \{0,2\} \\ -\frac{1}{2} & \text{if } T = \{0,1,2\} \\ 0 & \text{otherwise} \end{cases}.$$

(This generalizes the correlations computed in Example 7.2 in the main course.)

In linear cryptanalysis one uses a high correlation to get information about certain bits of the key. We shall see this work in an example.

**Example 6.2.** For $k \in \mathbb{F}_2^{12}$ let $e_k : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ be the $Q$-block cipher, as defined in Example 8.4. Then $e_k\big((v, w)\big) = (v', w')$ where

$$v' = w + S\big(v + S(w + k^{(1)}) + k^{(2)}\big).$$

Recall that $k^{(1)} = (k_0, k_1, k_2, k_3)$ and $k^{(2)} = (k_4, k_5, k_6, k_7)$.

Example 6.1 suggests looking at $\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2\}})$. (See the optional question on Problem Sheet 9 for the theoretical justification for this.) We have $(L_{\{0\}} \circ e_k)\big((v, w)\big) = L_{\{0\}}\big((v', w')\big) = v'_0$ and $L_{\{2\}}\big((v, w)\big) = v_2$.

*Exercise:* using that $k_0^{(1)} = k_0, k_1^{(1)} = k_1, k_2^{(1)} = k_2$ and $k_2^{(2)} = k_6$, check that

$$v_0' = v_2 + (w_1 + k_1)(w_2 + k_2) + k_0 + k_6.$$

When we compute $\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2\}})$ by averaging over all $(v, w) \in \mathbb{F}_2^8$, the values of $k_1$ and $k_2$ are irrelevant. For instance, if both are 0 we average $(-1)^{w_1 w_2}$ over all four $(w_1, w_2) \in \mathbb{F}_2^2$ to get $\frac{1}{2}$; if both are 1 we average $(-1)^{(w_1+1)(w_2+1)}$, seeing the same summands in a different order, and still getting $\frac{1}{2}$. Hence

$$\begin{aligned}
\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2\}}) &= \frac{1}{2^8} \sum_{(v,w) \in \mathbb{F}_2^8} (-1)^{v_2 + w_1 w_2 + k_0 + k_6} (-1)^{v_2} \\
&= \frac{1}{2^8} \sum_{(v,w) \in \mathbb{F}_2^8} (-1)^{w_1 w_2 + k_0 + k_6} \\
&= (-1)^{k_0 + k_6} \frac{1}{4} \sum_{w_1, w_2 \in \{0,1\}} (-1)^{w_1 w_2} \\
&= \tfrac{1}{2} (-1)^{k_0 + k_6}.
\end{aligned}$$

We can estimate this correlation from a collection of plaintext/ciphertext pairs $(v, w), (v', w')$ by computing $(-1)^{v_0' + v_2}$ for each pair. We get

$$\begin{array}{ll}
(-1)^{k_0 + k_6} & \text{with probability } \frac{3}{4} \\
-(-1)^{k_0 + k_6} & \text{with probability } \frac{1}{4}
\end{array}$$

so the average is the correlation $\frac{1}{2}(-1)^{k_0 + k_6}$ which tells us $k_0 + k_6$.

Using our collection of plaintext/ciphertext pairs we can also estimate

$$\begin{aligned}
\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2,5\}}) &= \tfrac{1}{2}(-1)^{k_0 + k_6 + k_1} \\
\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2,6\}}) &= \tfrac{1}{2}(-1)^{k_0 + k_6 + k_2}
\end{aligned}$$

and so learn $k_1$ and $k_2$ as well as $k_0 + k_6$. (You are asked to show this on Problem Sheet 9.) There are similar high correlations of $\frac{1}{2}$ for output bit 1. Using these one learns $k_2$ and $k_3$ as well as $k_1 + k_7$.

**Exercise 6.3.** Given $k_0 + k_6, k_1 + k_7, k_1, k_2, k_3$, how many possibilities are there for the key in the $Q$-block cipher?

This exercise shows that linear cryptanalysis gives a sub-exhaustive attack on the $Q$-block cipher. It is more powerful than the differential attack seen in the main course.

The attack by differential cryptanalysis required chosen plaintexts. The attack by linear cryptanalysis works with any observed collection of plaintext/ciphertext pairs. It is therefore more widely applicable, as well as more powerful.