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.

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 5pm (ALT3), Friday 11am (McCrea 2-01), Friday 4pm (BLT2).

Extra lecture for MSc students doing MT5462: Thursday 1pm (MFoxSem).

Office hours in McCrea LGF 0-25: Tuesday 3.30pm, Wednesday 11am, Thursday 11.30am (until 12.30pm) or by appointment.

Relevant seminar: The Information Security Group Seminar is at 11am Thursdays. To subscribe to the mailing list go to: `www.lists.rhul.ac.uk/mailman/listinfo/isg-research-seminar`.

Date: First term 2019/20.

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 as well: for example, a fundamental building block in AES (Advanced Encryption Standard) is the inversion map $x \mapsto x^{-1}$ on the non-zero elements of 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.

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 1.2.

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.

If you are familiar with basic group theory, it will be helpful to note that (1)–(4) say that $\mathbb{F}$ is an abelian group under addition, and that (5)–(8) say 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	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

$\times$	1	α	$1+\alpha$
1	1	α	$1+\alpha$
α	α	$1+\alpha$	1
$1+\alpha$	$1+\alpha$	1	α

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

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

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

Definition 1.4. If $f(x) = a_0 + a_1 x + a_2 + \cdots + a_m x^m$ where $a_m \neq 0$, then we say that m is the *degree* of the polynomial f, and write $\deg f = m$. The degree of the zero polynomial is, by convention, -1. We say that a_0 is the *constant term* and a_m is the *leading 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 left undefined.

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`, with `Modulus -> p` for finite fields.

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

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

Conversely, given t values, there is a polynomial of degree at most t taking these values at any t distinct specified points. This has a nice constructive 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

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

Later we shall use polynomials in multiple variables with coefficients in $\mathbb{F}_2$ to describe cryptographic primitives.

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 Person 10 whispers $M + s_1 + s_2 + \cdots + s_{10}$ to Person 1. Person 1 then subtracts M and tells 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 $M + s_1 + s_2$ by subtracting $M + s_1$. 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, labelled 1 up to n.

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, to share the secret $s \in \mathbb{F}_p$, 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 $c_1, \ldots, c_n$ and the identities of Persons 1 up to 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 $a_1 = 6$ 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

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

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 $f_{s'}(x)$ 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 can 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.

Exercise 2.8. Suppose Trevor shares $s \in \mathbb{F}_p$ across n computers using the Shamir scheme with threshold t. He chooses the first t computers. They are instructed to exchange their shares; then each computes s and sends it to Trevor. Unfortunately Malcolm has compromised computer 1. Show that Malcolm can both learn s and trick Trevor into thinking his secret is an $s' \in \mathbb{F}_p$ of his choice. (Assume that, thanks to a network delays, it is plausible that computer 1 sends its share after receiving the shares from the other $t-1$ computers.)

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

Example 2.9. The root key for DNSSEC, part of web of trust that guarantees an IP connection really is to the claimed end-point, and not to 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 2.10. 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 $(f(1), \ldots, f(n))$ and $(g(1), \ldots, g(n))$.

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

(b) Assuming that $n \geq 2t$, how can Trevor secret share $rs \bmod p$?

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

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

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

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 $n-t+1$.

We have worked over a finite field of prime size 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 are defined using the finite field $\mathbb{F}_{2^8}$.

3. Introduction to Boolean functions

Definition and first examples. Recall that $\mathbb{F}_2 = \{0, 1\}$ is the finite field of size 2 whose elements are the *bits* 0 and 1. As usual, $+$ denotes addition in $\mathbb{F}_2$ or in $\mathbb{F}_2^n$. We number positions in $\mathbb{F}_2^n$ from 0, so a typical tuple is $(x_0, x_1, \ldots, x_{n-1})$.

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

For example, $f(x,y,z) = xyz + x$ is a Boolean function of the three variables x, y and z, such that $f(1,0,0) = 0 + 1 = 1$ and $f(1,1,1) = 1 + 1 = 0$. We shall see that Boolean functions are very useful for describing the primitive building blocks of modern stream and block ciphers.

Exercise 3.2. What is a simpler form for $x^2y + xz + z + z^2$?

Exercise 3.3. Let $\mathrm{maj}(x,y,z) = xy + yz + zx$ where, as usual, the coefficients are in $\mathbb{F}_2$. Show that

$$\mathrm{maj}(x,y,z) = \begin{cases} 0 & \text{if at most one of } x,y,z \text{ is } 1 \\ 1 & \text{if at least two of } x,y,z \text{ are } 1. \end{cases}$$

We call $\mathrm{maj} : \mathbb{F}_2^3 \to \mathbb{F}_2$ the *majority vote function*. It is a 3-variable Boolean function.

Motivation. A modern block cipher has plaintexts and ciphertexts $\mathbb{F}_2^n$ for some fixed n. The encryption functions are typically defined by composing carefully chosen cryptographic primitives over a number of *rounds*. We give two motivating examples below.

Example 3.4.

(1) Each round of the widely used block cipher AES is of the form $(x,k) \mapsto G(x) + k$ where $+$ is addition in $\mathbb{F}_2^{128}$, $x \in \mathbb{F}_2^{128}$ is the input to the round (derived ultimately from the plaintext) and $k \in \mathbb{F}_2^{128}$ is a 'round key' derived from the key. The most important cryptographic primitive in the function $G : \mathbb{F}_2^{128} \to \mathbb{F}_2^{128}$ is inversion in the finite field $\mathbb{F}_{2^8}$. The inversion function is highly non-linear and hard to attack. Just for fun, the 256 values of the boolean function sending 0 to 0 and a non-zero x to the bit in position 0 of x^{-1} are shown below, for one natural order on $\mathbb{F}_{2^8}$.

```
0110101101100111000111010110100000001110110010000010011000101111
1011111110110111101000110000101100111001011111111111010000001010
1010010010111010000100000001010101001101000000010000111101100110 01
1011000111101000010111000101100111010011001110011100001010101010.
```

It is highly unlikely that you will see any obvious pattern! As one sign of the apparent randomness, there are 128 zeros, 128 ones, and each pair 01, 10, 11 appears exactly 64 times. Later we shall prove that the inversion function is secure against difference attacks.

The function for bit number 1 in the key addition is

$$(y_0, y_1, \ldots, y_{127}, k_0, k_1, \ldots, k_{127}) \mapsto y_1 + k_1.$$

We shall see that such linear functions are very weak cryptographically taken on their own, but are very useful when combined with non-linear functions such as G and inversion.

(2) In the block cipher SPECK proposed by NSA in June 2013, the non-linear primitive is modular addition in $\mathbb{Z}/2^m\mathbb{Z}$. As a 'toy' version we take $m = 8$; in practice m is at least 16 and usually 64. Identify $\mathbb{F}_2^8$ with $\mathbb{Z}/2^8\mathbb{Z}$ by writing numbers in their binary form, as on the preliminary problem sheet. For instance, $13 \in \mathbb{Z}/2^8\mathbb{Z}$ has binary form 0000 1101 (the space is just for readability) and

$$1010\,1010 \boxplus 0000\,1111 = 1011\,1001$$
$$1000\,0001 \boxplus 1000\,0001 = 0000\,0010$$

corresponding to $170 + 15 = 185$ mod 256 and $129 + 129 = 2$ mod 256. Modular addition is a convenient operation because it is very fast on a computer, but it has some cryptographic weaknesses. In SPECK it is combined with other functions in a way that appears to give a very strong and fast cipher.

One sign that modular addition is weak is that the low numbered bits are 'close to' linear functions. We make this precise in §6 on linear cryptanalysis. For example

$$\begin{aligned}(\ldots, x_2, x_1, x_0) &\boxplus (\ldots, y_2, y_1, y_0) \\ &= (\ldots, x_2 + y_2 + c_2, x_1 + y_1 + x_0y_0, x_0 + y_0)\end{aligned}$$

where c_2 is the carry into position 2, defined using the majority vote function by $c_2 = \mathrm{maj}(x_1, y_1, x_0y_0)$. Unless both x_0 and y_0 are 1, bit 1 is $x_1 + y_1$, a linear function of $(\ldots, x_2, x_1, x_0)$ and $(\ldots, y_2, y_1, y_0)$. By Exercise 4.4, output bit 2 is given by the more complicated polynomial

$$x_2 + y_2 + x_1y_1 + x_0x_1y_0 + x_0y_0y_1.$$

This formula can be used for part of Question 5 on Problem Sheet 3: it is the algebraic normal form of the boolean function for bit 2 in modular addition.

Truth tables and disjunctive normal form. A boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be defined by its *truth table*, which records for each $x \in \mathbb{F}_2^n$ its image $f(x)$. For example, the boolean functions $\mathbb{F}_2^2 \to \mathbb{F}_2$ of addition and multiplication are shown below:

x	y	$x+y$	xy	$x \wedge y$	$x \vee y$	$x \implies y$
0	0	0	0	F	F	
0	1	1	0	F	T	
1	0	1	0	F	T	
1	1	0	1	T	T	

It is often useful to think of 0 as false and 1 as true. Then xy corresponds to $x \wedge y$, the logical 'and' of x and y, as shown above. The logical 'or' of x and y is denoted $x \vee y$.

Exercise 3.5. Use the true/false interpretation to complete the columns for $x \implies y$. Could you convince a sceptical friend that false statement imply true statements?

Example 3.6. The Toffoli function is a 3-variable boolean function important in quantum computing. It can be defined by

$$\text{toffoli}(x_0, x_1, x_2) = \begin{cases} x_0 & \text{if } x_1 x_2 = 0 \\ \overline{x_0} & \text{if } x_1 x_2 = 1. \end{cases}$$

Here $\overline{x}$ denotes the bitflip of x, defined by $\overline{0} = 1$ and $\overline{1} = 0$. (You will have seen this if you did the Preliminary Problem Sheet.) In the true/false interpretation $\overline{F} = T$ and $\overline{T} = F$. It is shown below. The two final columns show the f_J functions defined later.

	x_2	x_1	x_0	$\text{maj}(x_0,x_1,x_2)$	$\text{toffoli}(x_0,x_1,x_2)$	$f_{\{0\}}$	$f_{\{0,2\}}$
$\varnothing$	0	0	0	0	0	0	0
$\{0\}$	0	0	1	0	1	1	0
$\{1\}$	0	1	0	0	0	0	0
$\{0,1\}$	0	1	1	1	1	0	0
$\{2\}$	1	0	0	0	0	0	0
$\{0,2\}$	1	0	1	1	1	0	1
$\{1,2\}$	1	1	0	1	1	0	0
$\{0,1,2\}$	1	1	1	1	0	0	0

The sets on the left record which variables are true. For example, the majority vote function is true on the rows labelled by the sets of sizes 2 and 3, namely, $\{0,1\}, \{0,2\}, \{1,2\}, \{1,2,3\}$, and false on the other rows.

Given a subset J of $\{0,\ldots,n-1\}$ we define $f_J : \mathbb{F}_2^n \to \mathbb{F}_2$ by

$$f_J(x) = \bigwedge_{j \in J} x_j \wedge \bigwedge_{j \notin J} \overline{x}_j.$$

In words, f_J is the n-variable boolean function whose truth table has a unique 1 (or true) in the row labelled J. For instance $f_{\{0\}}(x_0,x_1,x_2) = x_0 \wedge \overline{x}_1 \wedge \overline{x}_2$ and $f_{\{0,2\}}(x_0,x_1,x_2) = x_0 \wedge \overline{x}_1 \wedge x_2$ are shown above.

Exercise 3.7.

(i) For what set J do we have

$$\text{toffoli} = f_{\{0\}} \vee f_{\{0,1\}} \vee f_{\{0,2\}} \vee f_J?$$

(ii) Express the majority vote function in the form above.

(iii) Find a way to complete the right-hand side in

$$\text{maj}(x) = (x_0 \wedge x_1 \wedge \overline{x}_2) \vee (x_0 \wedge \overline{x}_1 \wedge x_2) \vee (\overline{x}_0 \wedge x_1 \wedge x_2) \vee (\ldots).$$

Theorem 3.8 (Disjunctive Normal Form). *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a boolean function.*

(i) *Suppose that the truth table of f has 1 in the rows labelled by the sets J for $J \in \mathcal{T}$. Then*
$$f = \bigvee_{J \in \mathcal{T}} f_J.$$

(ii) *If $\mathcal{T} \neq \mathcal{T}'$ then $\bigvee_{J \in \mathcal{T}} f_J \neq \bigvee_{J \in \mathcal{T}'} f_J$.*

This theorem says that every boolean function f has a unique *disjunctive normal form* $\bigvee_{J \in \mathcal{T}} f_J$, for a suitable set $\mathcal{T}$. (Disjunction means logical or', i.e. $\bigvee$.) By convention, the empty disjunction is false: $\bigvee_{J \in \varnothing}(x) = 0$ for all $x \in \mathbb{F}_2^n$.

Corollary 3.9. *There are 2^{2^n} n-variable boolean functions.*

Exercise 3.10. By Corollary 3.9, there are 16 truth tables of 2-variable boolean functions. Using the true/false notation, the 8 for which $f(F,F) = F$ are shown below. What is a suitable label for the rightmost column? What are the disjunctive normal forms of these 8 functions? What is a concise way to specify the remaining 8 functions?

	x_1	x_0	$x_0 \vee x_1$	x_0	x_1	$x_0 + x_1$	$x_0 \wedge x_1$	$x_0 \wedge \overline{x}_1$	$\overline{x}_0 \wedge x_1$	??
$\varnothing$	F	F	F	F	F	F	F	F	F	F
$\{0\}$	F	T	T	T	F	T	F	T	F	F
$\{1\}$	T	F	T	F	T	T	F	F	T	F
$\{0,1\}$	T	T	T	T	T	F	T	F	F	F

Algebraic normal form. In $\mathbb{F}_2$ we have $0^2 = 0$ and $1^2 = 1$. Therefore the Boolean functions $f(x_1) = x_1^2$ and $f(x_1) = x_1$ are equal. Hence, as seen in Exercise 3.2, multivariable polynomials over $\mathbb{F}_2$ do not need squares or higher powers of the variables. Similarly, since $2x_1 = 0$, the only coefficients needed are the bits 0 and 1. For instance, $x_0 + x_0x_2^2x_3^3 + x_0^2 + x_2x_3$ is the same Boolean function as $x_2x_3 + x_0x_2x_3$.

Given $I \subseteq \{0, 1, \ldots, n-1\}$, **[1 for 0 notation error, several later similar corrections have also been made]** let
$$x_I = \prod_{i \in I} x_i.$$
We say the x_I are *boolean monomials*. By definition (or convention if you prefer), $x_\varnothing = 1$. For example, $x_{\{1,2\}} = x_1x_2$. It is one of the three boolean monomial summands of $\mathrm{maj}(x_0, x_1, x_2) = x_0x_1 + x_1x_2 + x_2x_0$.

The functions f_J so useful for proving Theorem 3.8 have a particularly simple form as polynomials:
$$f_J(x) = \prod_{j \in J} x_j \prod_{j \notin J} \overline{x}_j.$$

Exercise 3.11. Define the 3-variable Boolean function

$$g(x_0, x_1, x_2) = \begin{cases} 1 & \text{if } x_0 = x_1 = x_2 \\ 0 & \text{otherwise.} \end{cases}$$

Express g as sum of boolean monomials. The negation of g is defined by $\overline{g} = \overline{g(x)}$. What is $\overline{g}$ as a sum of boolean monomials?

Similarly you can use the truth table on page 10 to express the Toffoli function and its negation as a sum of boolean monomials. It is only a small generalization of Exercise 3.11 to prove the following theorem.

Theorem 3.12. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be an n-variable Boolean function.*

(a) *There exist unique coefficients $b_J \in \{0,1\}$, one for each $J \subseteq \{1, \ldots, n\}$ such that*
$$f = \sum_{I \subseteq \{0,1,\ldots,n-1\}} b_J f_J.$$

(b) *There exist unique coefficients $c_I \in \{0,1\}$, one for each $I \subseteq \{1, \ldots, n\}$, such that*
$$f = \sum_{I \subseteq \{0,1,\ldots,n-1\}} c_I x_I.$$

The expression for f in (b) is called the *algebraic normal form* of f.

As shorthand, we write $[x_I]f$ for the coefficient of x_I in the boolean function f. Thus $f = \sum_{I \subseteq \{0,1,\ldots,n-1\}} ([x_I]f) x_I$ is the algebraic normal form of f. It is possible to give an explicit formula for the coefficients $[x_I]f$. As motivation, consider the sums in the exercise below.

Exercise 3.13. Let $f(x, y, z) = 1 + x + xz + yz + xyz$. Let $g(x, y, z) = f(0, y, z) + f(1, y, z)$ and let

$$\begin{aligned} h(x, y, z) &= g(x, 0, z) + g(x, 1, z) \\ &= f(0, 0, z) + f(1, 0, z) + f(0, 1, z) + f(1, 1, z) \end{aligned}$$

Find the algebraic normal form of g and h. What is the connection between $g(0, 0, 0)$ and $h(0, 0, 0)$ and $[x]f$, $[xy]f$? How would you find $[xz]f$ and $[xyz]f$ by this method?

Proposition 3.14. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be an n-variable Boolean function. Then*

$$[x_I]f = \sum f(z_0, \ldots, z_{n-1})$$

where the sum is over all $z_0, \ldots, z_{n-1} \in \{0,1\}$ such that $\{j : z_j = 1\} \subseteq I$.

In outline, the proof given in lectures is as follows: we first prove the formula when $f = f_J$ for some J. We then use Theorem 3.12(a), that f is a sum of the f_J, to get the general case.

4. 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 an n-variable boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ we define $(-1)^f : \mathbb{F}_2^n \to \{-1, 1\}$ by $(-1)^f(x) = (-1)^{f(x)}$.

Definition 4.1. 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 summand $(-1)^{f(x)}(-1)^{g(x)}$ is 1 when $f(x) = g(x)$ and -1 when $f(x) = -g(x)$. Hence

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

where

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

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

Linear functions such as $f(x_0, x_1, x_2) = x_0 + x_1$ are weak cryptographically. So are functions such as $f(x_0, x_1, x_2) = x_0 + x_1 x_2$ that are highly correlated with linear functions. Given $T \subseteq \{0, 1, \ldots, n-1\}$, define $L_T : \mathbb{F}_2^n \to \mathbb{F}_2$ by

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

For example, $L_{\{i\}}(x_0, x_1, \ldots, x_{n-1}) = x_i$ returns the entry in position i and $L_\varnothing(x) = 0$ is the zero function.

Exercise 4.2. Find all the linear 3-variable boolean functions. Which 3-variable boolean functions are uncorrelated with the zero function?

Lemma 4.3. *The linear functions $\mathbb{F}_2^n \to \mathbb{F}$ are precisely the $L_T : \mathbb{F}_2^n \to \mathbb{F}_2$ for $T \subseteq \{0, 1, \ldots, n-1\}$. If $S, T \subseteq \{0, 1, \ldots, n-1\}$ then*

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

Example 4.4. Let $\operatorname{maj} : \mathbb{F}_2^3 \to \mathbb{F}_2$ be the majority vote function from Exercise . We have **[corrected off-by-one error]**

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

To generalize the previous example, we define an inner product on the vector space W 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 4.5.

(i) Let $\theta \in W$. 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 $x \in \mathbb{F}_2^n$.

(ii) Show that if $n = 2$ then W is 4-dimensional. What is $\dim W$ in general?

It is immediate from the definition that if f and g are n-variable boolean functions then

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

Theorem 4.6 (Discrete Fourier Transform).

(a) *The functions $(-1)^{L_T}$ for $T \subseteq \{0, 1, \ldots, n-1\}$ are an orthonormal basis for the vector space W of functions $\mathbb{F}_2^n \to \mathbb{R}$.*

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

$$\theta = \sum_{T \subseteq \{0,1,\ldots,n-1\}} \langle \theta, (-1)^{L_T} \rangle (-1)^{L_T}.$$

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

$$(-1)^f = \sum_{T \subseteq \{0,1,\ldots,n-1\}} \mathrm{corr}(f, L_T)(-1)^{L_T}.$$

We call (c) the 'Discrete Fourier Inversion Theorem'. The function $T \mapsto \mathrm{corr}(f, L_T) = \langle (-1)^f, (-1)^{L_T} \rangle$ is the *Discrete Fourier Transform* of f. For example, by Example 4.4, the Discrete Fourier Transform of the majority vote function is

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

The following corollary is known as Parseval's Theorem.

Corollary 4.7. *Let f be an n-variable boolean function. Then*

$$\sum_{T \subseteq \{0,1,\ldots,n-1\}} \mathrm{corr}(f, L_T)^2 = 1.$$

Since there are 2^n linear functions (corresponding to the 2^n subsets of $\{0, 1, \ldots, n-1\}$), it follows that any n-variable boolean function f has a squared correlation of at least $1/2^n$. Hence f has a correlation of at least $1/\sqrt{2^n}$ in absolute value with some linear function.

Example 4.8.

(1) Let $f(x_0, x_1, x_2) = x_0x_1x_2$. We have $\mathrm{corr}(f, L_\varnothing) = \frac{3}{4}$, $\mathrm{corr}(f, L_{\{0\}}) = \frac{1}{4}$, $\mathrm{corr}(f, L_{\{0,1\}}) = -\frac{1}{4}$ and $\mathrm{corr}(f, L_{\{0,1,2\}}) = \frac{1}{4}$. By Theorem 4.6(c) and symmetry, the Discrete Fourier Transform of f is

$$(-1)^f = \tfrac{3}{4} + \tfrac{1}{4} \sum_{\substack{T \subseteq \{0,1,2\} \\ T \neq \varnothing}} (-1)^{|T|-1}(-1)^{L_T}.$$

The squares of the correlations are $\frac{9}{16}$ and $\frac{1}{14}$ (7 times); as expected from Corollary 4.7, $(\frac{3}{4})^2 + 7(\frac{1}{4})^2 = 1$.

(2) *Exercise:* Consider the 2-variable boolean function $f(x_0, x_1) = x_0x_1$. Find its correlations with the four linear functions $L_\varnothing(x_0, x_1) = 1$, $L_{\{0\}}(x_0, x_1) = x_0$, $L_{\{1\}}(x_0, x_1) = x_1$, $L_{\{0,1\}}(x_0, x_1) = x_1 + x_2$ and deduce that

$$(-1)^{x_0x_1} = \tfrac{1}{2}(-1)^{L_\varnothing} + \tfrac{1}{2}(-1)^{L_{\{0\}}} + \tfrac{1}{2}(-1)^{L_{\{1\}}} - \tfrac{1}{2}(-1)^{L_{\{0,1\}}}$$

(3) Let $b(x_0, x_1, y_0, y_1) = x_0y_0 + x_1y_1$. We shall use MATHEMATICA to show that $\operatorname{corr}(b, L_T) = \pm\frac{1}{4}$ for every $T \subseteq \{0, 1, 2, 3\}$. By the remark following Corollary 4.7, this function achieves the cryptographic ideal of having all correlations as small (in absolute value) as possible.

An n-variable boolean function such as b above where the correlations all have absolute value $1/\sqrt{2^n}$ is called a *bent function*. Since correlations are rational numbers, they exist only for even n. Many different constructions have been found and applied in cryptography.

We end with a lemma that is often useful for computing correlations. For instance applied to $x_0y_0, \ldots, x_{m-1}y_{m-1}$, and using Example 4.8(2) for the correlations of x_0y_0, it says that the correlations for $x_0y_0 + \cdots + x_{m-1}y_{m-1}$ are all $\pm 1/2^m$. Thus this function is bent. The special case $m = 2$ was seen in Example 4.8(3). See the end of Part B of the main course for an application to the quadratic stream cipher.

Lemma 4.9 (Piling-up Lemma)**.** *Let f be an m-variable boolean function of $x_0, \ldots, x_{m-1}$ and let g be an n-variable boolean function of $y_0, \ldots, y_{n-1}$. Define $f + g$ by*

$$(f + g)(x_0, \ldots, x_{m-1}, y_0, \ldots, y_{n-1}) = f(x_0, \ldots, x_{m-1}) + g(y_0, \ldots, y_{n-1}).$$

Given $S \subseteq \{0, \ldots, m-1\}$ and $T \subseteq \{0, \ldots, n-1\}$, let $L_{(S,T)}(x, y) = L_S(x) + L_T(y)$. The $L_{(S,T)}$ are all linear functions of the $m + n$ variables and

$$\operatorname{corr}(f + g, L_{(S,T)}) = \operatorname{corr}(f, L_S) \operatorname{corr}(g, L_T).$$

Since time is pressing, the proof may be omitted. But since it is very short, we give it below.

Proof of Lemma 4.9. The first claim is immediate from Lemma 4.3, applied with $m + n$ variables. By the Discrete Fourier Transform (Theorem 4.6(c)) we have

$$(-1)^f = \sum_{S \subseteq \{0,\ldots,m-1\}} \operatorname{corr}(f, L_S)(-1)^{L_S}$$
$$(-1)^g = \sum_{T \subseteq \{0,\ldots,n-1\}} \operatorname{corr}(g, L_T)(-1)^{L_T}$$

Observe that by definition of $L_{(S,T)}$,

$$(-1)^{L_S(x_0,\ldots,x_{m-1})}(-1)^{L_T(y_0,\ldots,y_{n-1})} = (-1)^{L_{(S,T)}(x_0,\ldots,x_{m-1},y_0,\ldots,y_{n-1})}.$$

Therefore multiplying the discrete fourier transforms gives

$$(-1)^{f+g} = \sum_{S\subseteq\{0,\ldots,m-1\}}\ \sum_{T\subseteq\{0,\ldots,n-1\}} \operatorname{corr}(f,L_S)\operatorname{corr}(g,L_T)(-1)^{L_{(S,T)}}.$$

This is the Discrete Fourier Transform of $(-1)^{f+g}$. Taking the coefficient of $(-1)^{L_{(S,T)}}$ we get $\operatorname{corr}(f+g, L_{(S,T)}) = \operatorname{corr}(f,L_S)\operatorname{corr}(g,L_T)$. $\square$

5. Berlekamp–Massey algorithm

The Berlekamp–Massey algorithm finds the width and feedback polynomial of an LFSR of minimal width generating a given binary word $u_0 \ldots u_{n-1}$. It is faster than the linear algebra method seen in Question 3 of Sheet 5. If an LFSR generates $u_0 \ldots u_{n-1}$ then clearly it generates $u_0 \ldots u_{m-1}$ for all $m \le n$. Therefore the minimal width stays the same or goes up at each step of the algorithm.

Motivation. These examples can be checked using the MATHEMATICA notebook `LFSRs.nb` available from Moodle. Recall from after Exercise 6.9 in the main course that the feedback polynomial of an LFSR with taps T is $g_T(z) = 1 + \sum_{t\in T} z^t$.

Example 5.1. By Question 4 on Sheet 5, the sum u of the keystreams of the LFSR with taps $\{3,4\}$ and width 4 and the LFSR with taps $\{2,3\}$ and width 3, using keys 0001 and 001, has period 105.

$$u_i = (0,0,1,1,1,1,0,1,0,0,0,0,0,0,1,0,1,0,0,1,1,1,1,1,1,1,0,0,1,1,\ldots)$$
$$\scriptstyle 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$$

The table below shows the output of the Berlekamp–Massey algorithm (use `BerlekampMasseyFull[usQ4]` in `LFSRs.nb`) applied to the first n terms $u_0 \ldots u_{n-1}$ for $n \ge 6$. The final column is the m in Proposition 5.5; ignore it for now.

n	width	feedback polynomial	taps	m
6	3	$1+z$	$\{1\}$	2
9	4	$1+z+z^4$	$\{1,4\}$	6
10	6	$1+z+z^3$	$\{1,3\}$	9
11	6	$1+z^2+z^3+z^5$	$\{2,3,5\}$	9
≥ 13	7	$1+z^2+z^4+z^5+z^7$	$\{2,4,5,7\}$	12

The LFSR does not change for $n = 7, 8$ or 12.

For instance, the first 10 terms $u_0u_1 \ldots u_9$ are generated by the LFSR of width 6 with feedback polynomial $1+z+z^3$; its taps are $\{1,3\}$. Taking as the key $u_0u_1u_2u_3u_4u_5 = 001111$, the first 30 terms of the keystream are:

$$k_i = (0,0,1,1,1,1,0,1,0,0,1,1,1,0,1,0,0,1,1,1,0,1,0,0,1,1,1,0,1,0,\ldots)$$
$$u_i = (0,0,1,1,1,1,0,1,0,0,0,0,0,0,1,0,1,0,0,1,1,1,1,1,1,1,0,0,1,1,\ldots)$$
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

Since $k_{10} \neq u_{10}$, running the Berlekamp–Massey algorithm on the first 11 bits $u_0 \ldots u_9u_{10}$ gives a different LFSR. (The width stays as 6, but the taps change to $\{2,3,5\}$.) The new LFSR generates $u_0 \ldots u_9u_{10}u_{11}$, so is also correct for the first 12 bits. This is why there is no change for $n = 12$.

For all $n \geq 13$ the output of the algorithm is the LFSR of width 7 and feedback polynomial $1+z^2+z^4+z^5+z^7$; as suggested on the problem sheet, this may also be found by the method of annihilators.

Example 5.2. The first 30 bits output by the Geffe generator seen in Example 8.3 of the main course are:

$$u_i \quad 0,0,0,0,1,1,1,1,0,1,0,1,1,1,0,0,0,0,1,1,1,0,1,1,1,1,1,1,1,0$$
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

The output of the Berlekamp–Massey algorithm run on the first 20 bits is shown below. [**Corrected several errors on May 19th, probably algorithm was run on wrong keystream.**]

n	width	feedback polynomial	taps	m
8	5	$1+z$	$\{1\}$	4
9	5	$1+z+z^4$	$\{1,4\}$	4
14	9	$1+z+z^4+z^9$	$\{1,4,9\}$	13
18	9	$1+z+z^5+z^8+z^9$	$\{1,5,8,9\}$	13
19	10	$1+z^6+z^8$	$\{1,6,8\}$	18

Taking $n = 30$, an LFSR of width 15 is required; the set of taps is then $\{1,3,4,5,7,8,9,10,11,12\}$. We see that the minimal width of a LFSR generating the first n terms is about $n/2$. This is the typical case for a 'random' sequence. This, and the lack of any obvious pattern in the taps, show that the Geffe cipher is stronger cryptographically than the output of an LFSR.

Setup. Fix throughout a binary stream

$$u_0u_1u_2 \ldots.$$

Let $U_n(z) = u_0 + u_1z + \cdots + u_{n-1}z^{n-1}$ be the polynomial recording the first n terms. Recall from §1 that the degree of a non-zero polynomial $h(z)$ is its highest power of z.

Lemma 5.3. *The word $u_0u_1 \dots u_{n-1}$ is the output of the LFSR with width ℓ and taps T if and only if $U_n(z)g_T(z) = h(z) + z^n r(z)$ for some polynomials $h(z)$ and $r(z)$ with* $\deg h < \ell$.

Proof. Let $T \subseteq \{1, \dots, \ell\}$. We have $U_n(z)g_T(z) = h(z) + z^n r(z)$ where $\deg h < \ell$ if and only if the coefficient of z in the left-hand side is 0 for $\ell \le s < n$. Since $g_T(z) = 1 + \sum_{t \in T} z^t$, this holds if and only if

$$k_s + \sum_{t \in T_n} k_{s-t} = 0$$

for $\ell \le s < n$. Equivalently, the LFSR with taps T and width ℓ generates $u_0u_1 \dots u_{n-1}$. □

Example 5.4. Let $u = (0,0,1,1,1,1,0,1,0,0,0,0,0) = u_0 \dots u_{12}$ be the first 13 entries of the keystream in Example 5.1. The first 12 entries $u_0 \dots u_{11}$ are generated by the LFSR of width 6 with taps $\{2,3,5\}$. Correspondingly, by the 'if' direction of Lemma 5.3,

$$\begin{aligned}(z^2 + z^3 + z^4 + z^5 + z^7)&g_{\{2,3,5\}}(z)\\ &= (z^2 + z^3 + z^4 + z^5 + z^7)(1 + z^2 + z^3 + z^5)\\ &= z^2 + z^3 + z^5 + z^{12}\\ &= h(z) + z^{12} r(z)\end{aligned}$$

where $h(z) = z^2 + z^3 + z^5$ and $r(z) = 1$. This equation also shows that the 'only if' direction fails to hold when $n = 13$ since z^{12} is not of the form $z^{13} r(z)$. Correspondingly, by the 'only if' direction of Lemma 5.3, the LFSR generates $(0,0,1,1,1,1,0,1,0,0,0,0,\mathbf{1})$ rather than u.

Berlekamp–Massey step. At step n of the Berlekamp–Massey algorithm we have two LFSRs:

- An LFSR F_m of width ℓ_m with taps T_m, generating
$$u_0u_1 \dots u_{m-1}\overline{u}_m \dots .$$
- An LFSR F_n of width ℓ_n with taps T_n, where $n > m$, generating
$$u_0u_1 \dots u_{m-1}u_m \dots u_{n-1}.$$

Thus F_m is correct for the first m positions, and then wrong, since it generates $\overline{u}_m$ rather than u_m. If F_n generates $u_0u_1 \dots u_{m-1}u_m \dots u_{n-1}u_n$ then case (a) applies and the algorithm returns F_n. The next proposition deals with case (b), when F_n outputs $\overline{u}_n$ rather than u_n.

Proposition 5.5. *With the notation above, suppose that the LFSR F_n generates $u_0u_1 \dots u_{n-1}\overline{u}_n$. The LFSR with feedback polynomial*

$$z^{n-m} g_{T_m}(z) + g_{T_n}(z)$$

and width $\max(n - m + \ell_m, \ell_n)$ *generates $u_0u_1 \dots u_{n-1}u_n$.*

As a useful notation we write $[\geq d]$ for an unspecified polynomial with minimum power of z at least d. For instance $[\geq 5]$ could stand for z^5+z^8, or z^6, but not z^4. Several times below we use that $[\geq d]+[\geq d]=[\geq d]$.

Proof. For $r \in \{0,1,\ldots,n+1\}$, define $U_r(z)=\sum_{i=0}^{r-1} u_i z^i$. Thus $U_{n+1}(z)$ is the power series corresponding to $u_0u_1\ldots u_n$. Observe that

$$U_{n+1}(z)+z^n = U_n(z)+u_nz^n+z^n = U_n(z)+\overline{u}_nz^n.$$

Since F_n generates $u_0\ldots u_{n-1}\overline{u_n}$, Lemma 5.3 implies

$$\big(U_{n+1}(z)+z^n\big)g_{T_n}(z) = h_n(z) + (\geq n+1)$$

where $\deg h_n < \ell_n$. The same argument replacing n with m shows that

$$(U_{m+1}(z)+z^m)g_{T_m}(z) = h_m(z)+[\geq m+1]$$

where $\deg h_m < \ell_m$. Since $z^ng_{T_n}(z) = z^n+[\geq n+1]$, and similarly $z^mg_{T_m}(z)=z^m+[\geq m+1]$, we have

$$\begin{aligned} U_{n+1}(z)g_{T_n}(z) &= h_n(z)+z^n+[\geq n+1] \\ U_{m+1}(z)g_{T_m}(z) &= h_m(z)+z^m+[\geq m+1]. \end{aligned}$$

Hence

$$\begin{aligned} &U_{n+1}(z)\big(z^{n-m}g_{T_m}(z)+g_{T_n}(z)\big) \\ &\quad = z^{n-m}\big(U_{m+1}(z)+[\geq m+1]\big)g_{T_m}(z)+U_{n+1}(z)g_{T_n}(z) \\ &\quad = z^{n-m}U_{m+1}(z)g_{T_m}(z)+[\geq n+1]+U_{n+1}(z)g_{T_n}(z) \\ &\quad = \big(z^{n-m}h_m(z)+z^n+[\geq n+1]\big)+\big(h_n(z)+z^n+[\geq n+1]\big) \\ &\quad = z^{n-m}h_m(z)+h_n(z)+(\geq n+1). \end{aligned}$$

where the first equality uses $U_{n+1}(z)=U_{m+1}(z)+[\geq m+1]$. Note the cancellation of the two z^n terms. (Intuitively: two wrongs come together to make a right.) Since

$$\begin{aligned} \deg\big(z^{n-m}h_m(z)+h_n(z)\big) &< \max\big(n-m+\deg h_m(z), \deg h_n(z)\big) \\ &\leq \max(n-m+\ell_m, \ell_n), \end{aligned}$$

the 'if' direction of Lemma 5.3 now implies that $u_0\ldots u_{n-1}u_n$ is a keystream of the claimed LFSR. $\square$

Example 5.6. Take the keystream $k_0k_1\ldots k_9$ of length 10 shown below:

$$\begin{array}{cccccccccc} (1, & 1, & 1, & 0, & 1, & 0, & 1, & 0, & 0, & 0). \\ {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 2} & {\scriptstyle 3} & {\scriptstyle 4} & {\scriptstyle 5} & {\scriptstyle 6} & {\scriptstyle 7} & {\scriptstyle 8} & {\scriptstyle 9} \end{array}$$

The LFSR F_6 of width $\ell_6=3$ and taps $T_6=\{1,3\}$ generates the keystream

$$\begin{array}{cccccccccc} (1, & 1, & 1, & 0, & 1, & 0, & 0, & 1, & 1, & 1). \\ {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 2} & {\scriptstyle 3} & {\scriptstyle 4} & {\scriptstyle 5} & {\scriptstyle 6} & {\scriptstyle 7} & {\scriptstyle 8} & {\scriptstyle 9} \end{array}$$

The LFSR F_7 of width $\ell_7=4$ and taps $T_7=\{1,4\}$ generates the keystream

$$\begin{array}{cccccccccc} (1, & 1, & 1, & 0, & 1, & 0, & 1, & 1, & 0, & 0). \\ {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 2} & {\scriptstyle 3} & {\scriptstyle 4} & {\scriptstyle 5} & {\scriptstyle 6} & {\scriptstyle 7} & {\scriptstyle 8} & {\scriptstyle 9} \end{array}$$

Note that F_7 is wrong in position 7. Using Proposition 5.5, taking $m = 6$ and $n = 7$ we compute

$$\begin{aligned} z^{n-m} g_{T_m} + g_{T_n}(z) &= z^{7-6} g_{\{1,3\}}(z) + g_{\{1,4\}}(z) \\ &= z(1 + z + z^3) + (1 + z + z^4) \\ &= 1 + z^2. \end{aligned}$$

This is the feedback polynomial of the LFSR F_8 with taps $T_8 = \{2\}$ and width $\ell_8 = n - m + \ell_m = 7 - 6 + 3 = 4$. As expected this generates

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

correct for the first 8 positions. (And then wrong for u_8.) Although the only tap in $\{2\}$ is 2, we still have to take the width of F_8 to be 4 (or more), to get the first 8 positions correct.

Exercise 5.7. Continuing from the example, apply Proposition 5.5 taking $n = 8$, $m = 6$, and F_8 and F_6 as in Example 5.6. You should get the LFSR F_9 with taps $\{3, 5\}$ generating

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

which is the full keystream. The width is now $8 - 6 + 3 = 5$; since 5 is a tap, this is the minimum possible width for these taps.

We could also have used F_7 (wrong in position 7) as the 'deliberately wrong' LFSR in Exercise 5.7. Doing this we get instead the LFSR with taps $\{1, 5\}$, which generates $(1, 1, 1, 0, 1, 0, 1, 0, 0, 1)$, also correct for the first 9 positions. We choose F_6 to follow the algorithm specified below.

Berlekamp–Massey algorithm. Let c be least such that $u_c \neq 0$. The algorithm defines LFSRs $F_c, F_{c+1}, \ldots$ so that each F_n has width ℓ_n and taps T_n and generates the first n positions of the keystream: $u_0, \ldots, u_{n-1}$.

- [Initialization] Set $T_c = \varnothing$, $\ell_c = 0$, $T_{c+1} = \varnothing$ and $\ell_{c+1} = c + 1$. Set $m = c$.
- [Step] We have an LFSR F_n with taps T_n of width ℓ_n generating $u_0, \ldots, u_{n-1}$ and an LFSR F_m generating $u_0, \ldots, u_{m-1}, \overline{u}_m$.
  - (a) If F_n generates $u_0, \ldots, u_{n-1}, u_n$ then set $T_{n+1} = T_n$, $\ell_{n+1} = \ell_n$. This defines F_{n+1} with $F_{n+1} = F_n$. Keep m as it is.
  - (b) If F_n generates $u_0, \ldots, u_{n-1}, \overline{u}_n$, calculate

    $$g(z) = z^{n-m} g_{T_m}(z) + g_{T_n}(z)$$

    where, as usual, g_{T_m} and g_{T_n} are the feedback polynomials. Define T_{n+1} so that $g(z) = 1 + \sum_{t \in T_{n+1}} z^t$. Set

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

If $\ell_{n+1} > \ell_n$, update m to n, otherwise keep m as it is.

Thus m is updated if and only if the width increases in step (b).

Apart from how the width changes, this should all be expected from Proposition 5.5 and Example 5.6. Note that we need $\max T_{n+1} \leq \ell_{n+1}$ for the LFSR F_{n+1} to be well-defined. We prove this as part of Theorem 5.10.

Example 5.8. We apply the Berlekamp–Massey algorithm to the keystream $(1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1)$ from Example 5.6 extended by one extra bit $u_{10} = 1$. After initialization we have $T_0 = \varnothing$, $\ell_0 = 0$, $T_1 = \varnothing$, $\ell_1 = 1$. Case (a) applies in each step n for $n \in \{2, 4, 5, 9\}$. The table below shows the steps when case (b) applies.

n	T_n	ℓ_n	m	T_m	$n-m$	T_{n+1}	ℓ_{n+1}
1	$\varnothing$	1	0	$\varnothing$	1	$\{1\}$	1
3	$\{1\}$	1	0	$\varnothing$	3 **[corr.]**	$\{1, 3\}$	3
6	$\{1, 3\}$	3	3	$\{1\}$	3	$\{1, 4\}$	4
7	$\{1, 4\}$	4	6	$\{1, 3\}$	1	$\{2\}$	4
8	$\{2\}$	4	6	$\{1, 3\}$	2	$\{3, 5\}$	5
10	$\{3, 5\}$	5	8	$\{2\}$	2	$\{2, 3, 4, 5\}$	6

Exercise.

- Run the algorithm starting with step 1, in which you should define $T_2 = \{1\}$, and finishing with step 6, in which you should define $T_7 = \{1, 4\}$.
- Then check that steps 7 and 8 of the algorithm are exactly what we did in Example 5.6 and Exercise 5.7.
- At step 9 you should find that case (a) applies; check that step 10 finishes with the LFSR F_{11} of width $\ell_{11} = 6$ and taps $T_{11} = \{2, 3, 4, 5\}$, generating $u_0 u_1 \ldots u_{10}$.

Berlekamp–Massey theorem. To prove that the LFSRs defined by running the Berlekamp–Massey algorithm have minimal possible width we need the following lemma. The proof is not obvious, but if you think 'what can I possibly do using Lemma 5.3' you should find the main idea.

Lemma 5.9. *Let $n \geq \ell$. If an LFSR F of width ℓ generates the keystream $(u_0, u_1, \ldots, u_{n-1}, b)$ of length $n + 1$ then any LFSR F' generating the keystream $(u_0, u_1, \ldots, u_{n-1}, \overline{b})$ has width ℓ' where $\ell' \geq n + 1 - \ell$.*

Proof. Let $U(z) = u_0 + u_1 z + \cdots + u_{n-1}z^{n-1} + bz^n$. By Lemma 5.3, using a similar argument to the proof of Proposition 5.5, we have

$$U(z)g_T(z) = h(z) + [\geq n+1]$$
$$\big(U(z) + z^n\big)g_{T'}(z) = h'(z) + [\geq n+1]$$

for polynomials $h(z)$ and $h'(z)$ with $\deg h < \ell$ and $\deg h < \ell'$. Since $z^n g_{T'}(z) = z^n + [\geq n+1]$, the second equation implies

$$U(z)g_{T'}(z) = h'(z) + z^n + [\geq n+1].$$

We now use these equations to compute $U(z)g_T(z)g_{T'}(z)$ in two different ways:

$$U(z)g_T(z)g_{T'}(z) = h(z)g_{T'}(z) + [\geq n+1]$$
$$U(z)g_{T'}(z)g_T(z) = h'(z)g_T(z) + z^n g_T(z) + [\geq n+1].$$

Using that $z^n g_T(z) = z^n + [\geq n+1]$ and adding we find that

$$0 = h(z)g_{T'}(z) + h'(z)g_T(z) + z^n + [\geq n+1].$$

Since $\deg h(z)g_{T'}(z) \leq \deg h + \ell' < \ell + \ell'$ and $\deg h'(z)g_T(z) < \deg h' + \ell = \ell' + \ell$, the only way that z^n can cancel is if $\ell + \ell' > n$. This is equivalent to $\ell' \geq n + 1 - \ell$. □

Recall that step n of the Berlekamp–Massey algorithm returns an LFSR F_{n+1} with taps T_{n+1} and width ℓ_{n+1} generating $u_0 \ldots u_{n-1}u_n$.

Theorem 5.10. *With the notation above,* $\max T_{n+1} \leq \ell_{n+1}$. *Moreover ℓ_{n+1} is the least width of any LFSR generating $u_0, \ldots, u_{n-1}, u_n$.*

Proof of Theorem 5.10 **[Version 1: skip please!]** Set $\ell_0 = 0$ and $T_0 = \varnothing$. We work by induction on $n \in \mathbb{N}$. So our inductive hypothesis is that

> *Claim:* For each $r \leq n$, F_r is the minimum width LFSR generating $u_0u_1 \ldots u_{r-1}$. Moreover, whenever $r < n$ and $T_r \neq T_{r+1}$ or $\ell_{r+1} \neq \ell_r$ equality holds in Lemma 5.9, i.e. $\ell_{r+1} = \max(\ell_r, r + 1 - \ell_r)$.

Base case: let c be least such that $k_c \neq 0$. When $n = c$ or $n = c + 1$, by the initialisation step, $T_c = T_{c+1} = \varnothing$ and $\ell_c = 0$ and $\ell_{c+1} = c + 1$. Clearly these are the minimum possible widths. **[Correction: it's not vacuous]**

Inductive step: Suppose the claim holds for $n \in \mathbb{N}$. We must prove Theorem 5.10, as stated above for n, and the 'moreover' part of the claim.

For (a), if F_n generates $u_0 \ldots u_{n-1}u_n$ then, since no shorter LFSR generates $u_0 \ldots u_{n-1}$, F_n is the minimal length lFSR that generates the keystream $u_0 \ldots u_{n-1}u_n$. The taps and lengths do not change, so there is nothing more to check.

In (b), we suppose F_n generates $u_0 \ldots u_{n-1}\overline{u}_n$. By Proposition 5.5 we get a new LFSR F_{n+1} generating $u_0 \ldots u_{n-1}u_n$ with taps T_{n+1} and width ℓ_{n+1}. 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

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

By definition, $\ell_{n+1} = \max(\ell_n, n+1-\ell_n)$. So by the previous displayed equation $n+1-\ell_n = n+1-(m+1-\ell_m) = n-m+\ell_m$ and

$$(\ddagger) \qquad \ell_{n+1} = \max(\ell_n, n-m+\ell_m).$$

We now consider two cases.

(i) Suppose $n-m+\ell_m \leq \ell_n$. Since $z^{n-m}g_{T_m}(z)$ has degree $\leq n-m+\ell_m$ and $g_{T_n}(z)$ has degree $\leq \ell_n$, their sum $g_{T_{n+1}}(z)$ has degree $\leq \ell_n$. Hence $T_{n+1} \subseteq \{1,\ldots,\ell_n\}$. By induction, no LFSR of width $< \ell_n$ generates $u_0\ldots u_{n-1}$, so ℓ_n is the minimal width of an LFSR generating $u_0\ldots u_{n-1}u_n$. By (‡) and the hypothesis $n-m+\ell_m \leq \ell_n$, we have $\ell_{n+1} = \ell_n$ as required.

(ii) Suppose that $n-m+\ell_m > \ell_n$. Since the LFSR of width ℓ_n generates $u_0\ldots u_{n-1}\overline{u}_n$, by Lemma 5.9, the minimum width of an LFSR generating $u_0\ldots u_{n-1}u_n$ is at least $n+1-\ell_n$. By (‡) and the hypothesis $n-m+\ell_m > \ell_n$, we have $\ell_{n+1} = n-m+\ell_m$. Hence $n-m+\ell_m = n-m+(m+1-\ell_n) = n+1-\ell_n$. Therefore $\ell_{n+1} = n+1-\ell_n$, as required.

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

[I now see that there is a simpler, and I find more intuitive, proof of the result we need. Base case is now correct (it's not vacuous).]

Proof. We work by induction on n.

Base case. let c be least such that $k_c \neq 0$. When $n = c$ or $n = c+1$, by the initialisation step, $T_c = T_{c+1} = \varnothing$ and $\ell_c = 0$ and $\ell_{c+1} = c+1$. Clearly these are the minimum possible widths.

Inductive step. By induction, ℓ_n is the minimum width of an LFSR generating $u_0u_1\ldots u_{n-1}$. Hence any LFSR generating $u_0u_1\ldots u_{n-1}u_n$ has width $\geq \ell_n$. By Lemma 5.9, the minimum width of an LFSR generating $u_0u_1\ldots u_{n-1}u_n$ is at least $n+1-\ell_n$. Therefore the minimum width of an LFSR generating $u_0u_1\ldots u_{n-1}u_n$ is at least

$$\max(\ell_n, n+1-\ell_n).$$

By definition, this is ℓ_{n+1}. Therefore, provided F_{n+1} is well-defined, that is $\max T_{n+1} \leq \ell_{n+1}$, it has the minimum possible width.

Suppose case (a) holds. By definition, $T_{n+1} = T_n$ and $\ell_{n+1} = \ell_n$. Hence $\max T_{n+1} = \max T_n \leq \ell_n = \ell_{n+1}$ as required. Suppose case (b) holds. Since the most recent width change was at step m, we have $\ell_{m+1} > \ell_m$, and since, by definition, $\ell_{m+1} = \max(\ell_m, m+1-\ell_m)$ we have $\ell_{m+1} = m+1-\ell_m$. Therefore $\ell_n = \ell_{n-1} = \ldots = \ell_{m+1} = m+1-\ell_m$ and

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

The taps T_{n+1} are defined by

$$1 + \sum_{t\in T_{n+1}} z^t = z^{n-m}g_{T_m}(z) + g_{T_n}(z).$$

Here $z^{n-m}g_{T_m}(z)$ has degree at most $n - m + \ell_m$, which is $n + 1 - \ell_n$ by the displayed equation above, and $g_{T_n}(z)$ has degree at most ℓ_n. Since, by definition, $\ell_{n+1} = \max(\ell_n, n + 1 - \ell_n)$, we have $\max T_{n+1} \leq \ell_{n+1}$. $\square$

The *linear complexity* of a word $u_0u_1 \ldots u_{n-1}$ is the minimal width of an LFSR that generates it. Modern stream ciphers aim to generate keystreams with high linear complexity. For example, take the m-quadratic stream cipher from Example 8.5. If $m = 1$ the keystream $u_0u_1 \ldots u_{29}$ for the key pair $k = 10101$ and $k' = 101010$ is

$$(1,0,1,0,1,0,0,0,0,1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,0,0,1,0,0,1).$$

The table below shows the linear complexity of the first n bits of the keystream for small n and m.

$m\backslash n$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	2	2	2	2	5	5	5	5	5	5	5	5	5
2	0	2	2	2	2	2	5	5	5	5	5	5	5	5	5
3	0	0	0	4	4	4	4	4	4	6	6	6	6	6	6
4	0	0	0	0	0	7	7	7	7	7	7	7	7	7	8
5	0	0	0	0	5	5	5	5	5	5	5	7	7	7	8

Some interesting features can be seen for larger lengths: for instance the linear complexity when $m = 1$ jumps from 5 for $n = 20$ to 16 for $n = 21$. For $n = 5$ the linear complexity is about $n/2$; this is the expected linear complexity of a random sequence of bits.

Extra. 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).

Example 5.11. The Berlekamp–Massey algorithm (for arbitrary fields) is implemented in the MATHEMATICA notebook `LFSRs.nb`. Try

```
BerlekampMasseyFull[{1,1,1,0,1,0,1,0,0,0,1}] // TF
```

to check Example 5.8. Each line of the output gives a pair

$$\big((m, \ell_m, e_m, g_{T_m}(z)), (n, \ell_n, e_n, g_{T_n}(z))\big).$$

Here e_m and e_n are the errors on bits u_m and u_n; for the binary case, $e_m = 1$ for all relevant m (since the LFSR changed) and $u_n = 1$ if and only if Step (b) applies. You can read the taps T_m and T_n off from the feedback polynomials. Using this you should be able to translate the MATHEMATICA output into the table in Example 5.8. To see an example where it finds a linear recurrence for an integer sequence try

```
BerlekampMasseyFull[{1,1,2,3,5,8,13,21,34},0] // TF
```

Now try instead the sequence $1, 1, 2, 3, 4, 6, 9, 13, 19, 28$. What do you expect is the next term?

6. Linear cryptanalysis

In §4 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.

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 $(G \circ F)(x) = G(F(x))$.

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 9.5 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

$$\begin{aligned}(L_{\{0\}} \circ S)\big((x_0, x_1, x_2, x_3)\big) &= L_{\{0\}}(x_2, x_3, x_0 + x_1x_2, x_1 + x_2x_3) \\ &= x_2 \\ &= L_{\{2\}}\big((x_0, x_1, x_2, x_3)\big).\end{aligned}$$

Hence $L_{\{0\}} \circ S = L_{\{2\}}$. By Lemma 4.3,

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

(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

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

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

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

We choose v' rather than w' since v' depends only on the first two round keys. 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 considering $\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2\}})$. 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

$$v'_0 = v_2 + (w_1 + k_1)(w_2 + k_2) + k_0 + k_6.$$

By definition

$$\begin{aligned}
\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2\}}) &= \frac{1}{2^8} \sum_{(v,w)\in\mathbb{F}_2^8} (-1)^{v_2+(w_1+k_1)(w_2+k_2)+k_0+k_6}(-1)^{v_2} \\
&= \frac{1}{2^8}(-1)^{k_0+k_6} \sum_{(v,w)\in\mathbb{F}_2^8} (-1)^{(w_1+k_1)(w_2+k_2)} \\
&= \frac{2^6}{2^8}(-1)^{k_0+k_6} \sum_{w_1,w_2\in\mathbb{F}_2} (-1)^{(w_1+k_1)(w_2+k_2)}
\end{aligned}$$

When we compute the sum, the values of k_1 and k_2 are irrelevant. For instance, if both are 0 we average $(-1)^{w_1w_2}$ 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

$$\begin{aligned}
\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2\}}) &= \frac{1}{2^8}(-1)^{k_0+k_6} \sum_{(v,w)\in\mathbb{F}_2^8} (-1)^{w_1w_2} \\
&= (-1)^{k_0+k_6}\frac{1}{4} \sum_{w_1,w_2\in\{0,1\}} (-1)^{w_1w_2} \\
&= \tfrac{1}{2}(-1)^{k_0+k_6}.
\end{aligned}$$

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

$$\begin{aligned}
(-1)^{k_0+k_6} &\quad \text{with probability } \tfrac{3}{4} \\
-(-1)^{k_0+k_6} &\quad \text{with probability } \tfrac{1}{4}
\end{aligned}$$

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 difference attack seen in the main course. Moreover, this attack required chosen plaintexts, rather than the observed collection of plaintext/ciphertext pairs used here. It is therefore more widely applicable.

In the attack on the Q-Block Cipher we saw that the correlation depended on the key only by a sign. This is because key addition, as is almost universally the case for block ciphers, was done in $\mathbb{F}_2^n$.

Lemma 6.4. *Fix $k \in \mathbb{F}_2^n$. Define $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ by $F(x) = x + k$. Then*

$$\operatorname{corr}(L_S \circ F, L_T) = \begin{cases} (-1)^{L_S(k)} & \text{if } S = T \\ 0 & \text{if } S \neq T. \end{cases}$$

Another very useful result gives correlations through the composition of two functions.

Proposition 6.5. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be functions. For $S, T \subseteq \{0, 1, \ldots, n-1\}$,*

$$\operatorname{corr}(L_S \circ G \circ F, L_T) = \sum_{U \subseteq \{0,1,\ldots,n-1\}} \operatorname{corr}(L_S \circ G, L_U) \operatorname{corr}(L_U \circ F, L_T).$$

Example 6.6.

(1) Take $G(x_0, x_1) = (x_0, x_0x_1)$. The matrix of correlations, with rows and columns labelled $\varnothing$, $\{0\}$, $\{1\}$, $\{0,1\}$ is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

(2) By Lemma 6.4, the matrix for $(x_0, x_1) \mapsto (x_0 + 1, x_1)$ is diagonal, with entries $1, -1, 1, 1$.

(3) Hence $H(x_0, x_1) = (x_0 + 1, x_0x_1 + x_1) = (\overline{x}_0, \overline{x}_0x_1)$ has correlation matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{pmatrix}.$$

We end by applying Proposition 6.5 to the S-box in the Q-block cipher. Let $F : \mathbb{F}_2^3 \to \mathbb{F}_2^3$ be the S-box in the 3 bit version of the Q-block cipher, so $F\big((x_0, x_1, x_2)\big) = (x_1, x_2, x_0 + x_1x_2)$. The matrix below shows the correlations,

$$\begin{pmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \frac{1}{2} & \cdot & \frac{1}{2} & \cdot & \frac{1}{2} & \cdot & -\frac{1}{2} \\ \cdot & \frac{1}{2} & \cdot & \frac{1}{2} & \cdot & -\frac{1}{2} & \cdot & \frac{1}{2} \\ \cdot & \frac{1}{2} & \cdot & -\frac{1}{2} & \cdot & \frac{1}{2} & \cdot & \frac{1}{2} \\ \cdot & -\frac{1}{2} & \cdot & \frac{1}{2} & \cdot & \frac{1}{2} & \cdot & \frac{1}{2} \end{pmatrix}$$

writing $\cdot$ for a 0 correlation, with subsets ordered

$$\varnothing, \{0\}, \{1\}, \{0,1\}, \{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}.$$

For example the first four rows show that tapping in positions $\varnothing$, $\{0\}$, $\{1\}$, or $\{0,1\}$ gives a linear function. By taking powers of this matrix we can compute correlations through any power of F.

In the lecture we will use MATHEMATICA to find the order of the (normal) four bit version of F.

The high correlations used in Example 6.2 were found by applying Proposition 6.5 to the Feistel functions and S-box in the Q-block cipher.