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 (MC219), Friday 4pm (MC219).

Extra lecture for M.Sc. students: Thursday 1pm (MC336).

Office hours in McCrea 240: Tuesday 3.30pm, Wednesday 10am, Thursday noon 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 2018/19.

OVERVIEW

Lecture 1

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. 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 1.2.

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.

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`.

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* **Lecture 2**

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

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

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

Multivariable polynomials. Polynomials in multiple variables are often useful for describing cryptographic primitives. For example, $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$ is a multivariable polynomial in the three variables x_1,x_2,x_3,x_4 and coefficients in $\mathbb{F}_2$.

Exercise 1.9. Let $a_1, a_2, a_3 \in \mathbb{F}_2$. Show that, as defined above,

$$f(a_1, a_2, a_3) = \begin{cases} 0 & \text{if at most one of the } a_i \text{ is } 1 \\ 1 & \text{if at least two of the } a_i \text{ are } 1. \end{cases}$$

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 M 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 $M + s_1 + s_2$ 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

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

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 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.* **Lecture 3**

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 any chosen $s' \in \mathbb{F}_p$.

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 $\big(f(1), \ldots, f(n)\big)$ and $\big(g(1), \ldots, g(n)\big)$.

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

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

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 $n - t + 1$.

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

Lecture 4

Definition and cryptographic motivation. Recall that $\mathbb{F}_2 = \{0, 1\}$ is the finite field of size 2. We refer to its elements as *bits*.

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

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 defined by the truth tables below.

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	0

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

It is sometimes useful to think of 0 as false (written F) and 1 as true (written T). Then multiplication corresponds to logical 'and'.

A typical modern cipher is defined by using boolean functions to define functions $\mathbb{F}_2^n \to \mathbb{F}_2^n$, and then composing them in a number of 'rounds'. We give two motivating examples below. To avoid eye-strain we write $(1, 0, 1, 1, 1, 0, 0, 1) \in \mathbb{F}_2^8$ as 1011 1001 and so on.

Example 3.2.

(1) As usual $+ : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ denotes vector space addition. For instance, if $n = 8$, then $1010\,1010 + 0000\,1111 = 1010\,0101$ and $1000\,0001 + 1000\,0001 = 0000\,0000$: note each sum can be computed bit-by-bit from the truth table for addition above.

Each round of the widely used block cipher AES is of the form $(x, k) \mapsto G(x) + k$ where $x \in \mathbb{F}_2^{256}$ is the input to the round (derived ultimately from the plaintext) and $k \in \mathbb{F}_2^{128}$ is derived from the key; the definition of $G : \mathbb{F}_2^{128} \to \mathbb{F}_2^{128}$ will be seen in Part C.

(2) In the block cipher FEAL, a critical 'mixing' function is modular addition in $\mathbb{Z}/2^8\mathbb{Z}$, denoted $\boxplus$. To define $\boxplus$ we 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,

$$10101010 \boxplus 00001111 = 10111001$$
$$10000001 \boxplus 10000001 = 00000010$$

corresponding to $170 + 15 = 185 \bmod 256$ and $129 + 129 = 2 \bmod 256$. Modular addition is a convenient operation because it is fast on a computer; unfortunately because of the way it is combined with the other functions in each round, FEAL is now famous only for the many ways in which it can be attacked.

Exercise 3.3. Motivated by FEAL, define $f : \mathbb{F}_2^4 \to \mathbb{F}_2$ by $f(x_1, x_0, y_1, y_0) = z_1$ where $x_1x_0 \boxplus y_1y_0 = z_1z_0 \bmod 4$. For instance, since $3 + 1 = 0 \bmod 4$ we have $11 \boxplus 01 = 00$ and so $f(1, 1, 0, 1) = 0$.

(a) Is f a Boolean function?
(b) Check that f is also defined by $f(x_1, x_0, y_1, y_0) = x_1 + y_1 + x_0y_0$.
(c) What is the connection with the arithmetic algorithm you learned at school?

Exercise 3.4. Complete the truth table for logical implication, writing F for 0 (false) and T for 1 (true).

x	y	$x \Longrightarrow y$
F	F	
F	T	
T	F	
T	T	

Lecture 5

Exercise 1.9 and Exercise 3.3 show that Boolean functions can be expressed in many different ways, not always obviously the same. In the remainder of this section we study 'normal forms' for boolean functions. Applications to cryptography will follow.

Lemma 3.5. *There are 2^{2^n} boolean function in n variables.*

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 multivariable polynomials 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, $x_1 + x_1x_2^2x_3^3 + x_1^2 + x_2x_3$ is the same Boolean function as $x_2x_3 + x_1x_2x_3$.

Exercise 3.6. Find a simple form for the product of $f(x_1, x_2, x_3) = x_1\overline{x_2}x_3$ and $\mathrm{maj}(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3x_1$. (Here $\overline{x_2} = 1 + x_2$ is the bit-flip of x_2, as defined on the Preliminary Problem Sheet.)

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 $\mathrm{maj}(x_1, x_2, x_3)$.

Example 3.7. The Toffoli gate is important in quantum computation. It takes 3 input qubits and returns 3 output qubits. Its classical analogue (which only returns one bit) 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'. We will find its algebraic normal form, first direct from this definition, and then from its truth-table.

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 $I \subseteq \{1, \ldots, n\}$, such that*

$$f = \sum_{I \subseteq \{1,\ldots,n\}} c_I x_I.$$

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, let $f(x_1, x_2, x_3) = x_1x_2 + x_3$ be the Toffoli function seen in Example 3.7. Then, by (c), $f(0, 0, 0) + f(1, 0, 0) + f(0, 1, 0) + f(1, 1, 0) = 0 + 0 + 0 + 1 = 1$ 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 $f(x_1, x_2, x_3) = x_1x_2x_3$?

Lecture 6

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*

$$f = \sum_{I \subseteq \{1,\ldots,n\}} c_I x_I.$$

Then

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

where the sum is over all $z_1, \ldots, z_n \in \{0, 1\}$ 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. Then the bitflip $\overline{x}$ corresponds to logical negation: $0 \leftrightarrow 1$ or $T \leftrightarrow F$.

Following the usual convention, we write $\wedge$ for 'logical and' (also called *conjunction*) and $\vee$ for 'logical or' (also called *disjunction*). In algebraic normal form, $x \wedge y = xy$ and $x \vee y = x + y + xy$. Note that $x \vee y$ is true if both x and y are true.

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

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

where

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

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

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

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

Example 3.13.

(a) The majority vote function $\mathrm{maj}(x_1, x_2, x_3)$ is true if and only if at two of x_1, x_2, x_3 are true. Therefore $\mathrm{maj}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_1 \wedge x_3)$. This is *not* in disjunctive normal form, but it is now only a short step to get

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

(b) We saw in Example 3.7 that the truth table for the Toffoli function $f(x_1, x_2, x_3) = x_1 x_2 + x_3$ 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, working down the truth table, as in the proof of Theorem 3.8, we get

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

(c) Let 1 be the constant Boolean function on 2 variables. We will use the same truth table trick to express 1 in disjunctive normal form.

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*

$$f = \bigvee_{J \in \mathcal{B}} f_J.$$

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

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,

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

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 $\overline{f}$ in disjunctive normal form and then negating it, using that if $J \subseteq \{1, \ldots, n\}$ then $\overline{f_J} = g_{J'}$ where $J' = \{k \in \{1, \ldots, n\} : k \notin J\}$.

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 $\overline{\text{maj}(x_1, x_2, x_3)} = f_\varnothing \vee f_{\{1\}} \vee f_{\{2\}} \vee f_{\{3\}}$ in disjunctive normal form and so

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

in conjunctive normal form. In words this says: $\text{maj}(x_1, x_2, x_3)$ is true if and only if at most one of x_1, x_2, x_3 is false.

4. Berlekamp–Massey Algorithm

Lecture 7

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 3 **[sorry, not 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. Recall from Definition 5.1(iii) that the keystream of the LFSR of width ℓ with taps $T \subseteq \{0, \ldots, \ell - 1\}$ generated by $(k_0, \ldots, k_{\ell-1}) \in \mathbb{F}_2^\ell$ is defined by $k_s = \sum_{t \in T} k_{s-\ell+t}$ for $s \geq \ell$. In the Berlekamp–Massey algorithm, it is more convenient to use the *backward taps*, defined by

$$\widetilde{T} = \{\ell - t : t \in T\}.$$

Note that $\widetilde{T} \subseteq \{1, \ldots, \ell\}$. With this notation,

$$(\dagger) \qquad k_s = \sum_{\widetilde{t} \in \widetilde{T}} k_{s-\widetilde{t}} \quad \text{for each } s \geq \ell.$$

We also need the *symmetric difference* of sets X and Y defined by

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

Equivalently, $T \triangle U$ is the elements lying in exactly one of T and U. The following lemma shows how symmetric differences arise when we combine LFSRs.

Lemma 4.1. *Let $T, U \subseteq \mathbb{N}_0$. Let f and g be the feedback functions for LFSRs with taps T and U, respectively, each of width at most ℓ. Then*

$$f\big((x_0, x_1, \ldots, x_{\ell-1})\big) + g\big((x_0, x_1, \ldots, x_{\ell-1})\big) = \sum_{s \in T \triangle U} x_s$$

*[**Corrected typo** $x \in T$] is the feedback function for an LFSR with taps $T \triangle U$ and backtaps $\widetilde{T} \triangle \widetilde{U}$.*

Berlekamp–Massey step. We fix throughout a sequence of bits $k_0, k_1, k_2, \ldots$.

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
$$k_0, k_1, \ldots k_{m-1}, \bar{k}_m \ldots;$$
- An LFSR F_n of width ℓ_n with taps T_n, where $n > m$, generating
$$k_0, k_1, \ldots, k_{m-1}, k_m, \ldots, k_{n-1}.$$

Thus F_m is correct for the first m positions, and then wrong, since it generates $\bar{k}_m$ rather than k_m. If F_n generates $k_0, \ldots, k_{m-1}, k_m, \ldots, k_{n-1}, k_n$, then the algorithm returns F_n; this is case (a). The next proposition is used to deal with case (b), when F_n is wrong after the first n positions.

Proposition 4.2. *With the notation above, suppose that the LFSR F_n generates $(k_0, k_1, \ldots, k_{n-1}, \overline{k}_n)$. Let $U = \{\widetilde{t} + n - m : \widetilde{t} \in \widetilde{T}_m\}$. Setting*

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

defines an LFSR F_{n+1} generating $(k_0, k_1, \ldots, k_{n-1}, k_n)$.

Proof. By (†)

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

Moreover, by our assumptions on F_m, and again by (†),

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

Since $\widetilde{T}_{n+1} = \widetilde{T}_n \triangle (U \cup \{n-m\})$, where $U = \{\widetilde{t} + n - m : \widetilde{t} \in \widetilde{T}_m\}$, Lemma 4.1 implies that

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

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 $(k_0, k_1, \ldots, k_{n-1}, k_n)$. □

Lecture 8

Example 4.3. Take the keystream $k_0 k_1 \ldots k_9$ of length 10 shown below:

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

The LFSR F_6 of width 3 and backtaps $\{1, 3\}$ generates the keystream

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

The LFSR F_7 of width 4 and backtaps $\{1, 4\}$ generates the keystream

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

Note that F_7 is wrong in position 7. Using Proposition 4.6, we take $U = \{\widetilde{t} + 7 - 6 : \widetilde{t} \in \widetilde{T}_6\} = \{2, 4\}$ and

$$\widetilde{T}_8 = \widetilde{T}_7 \triangle (U \cup \{7 - 6\}) = \{1, 4\} \triangle (\{2, 4\} \cup \{1\}) = \{2\}.$$

We obtain the LFSR F_8 with backtaps $\{2\}$ generating

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

Although the only backtap 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 4.4. Observe that F_8 is correct for the first 8 positions, up to $k_7 = 0$, then wrong. Apply Proposition 4.2 taking $n = 8$, $m = 6$, and F_8 and F_6 as in Example 4.3. You should get the LFSR F_9 with backtaps $\{3, 5\}$ generating

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

Since 5 is a backtap of F_9 we have to take its width to be 5 (or more).

We could also have used F_7 (wrong in position 7) as the 'deliberately wrong' LFSR in Exercise 4.4. Doing this we get instead the LFSR with backtaps $\{1, 5\}$, which generates $(1, 1, 1, 0, 1, 0, 1, 0, 0, 1)$, also correct to 9 positions. We choose F_6 to follow the algorithm specified below.

Berlekamp–Massey algorithm. Let c be least such that $k_c \neq 0$. The algorithm defines LFSRs $F_c, F_{c+1}, \ldots$ so that each F_n has width ℓ_n and backtaps $\widetilde{T}_n$ and generates the first n positions of the keystream: $k_0, \ldots, k_{n-1}$.

- [Initialization] Set $\widetilde{T}_c = \varnothing$, $\ell_c = 0$, $\widetilde{T}_{c+1} = \varnothing$ and $\ell_{c+1} = c + 1$. **[Corrected from** $\ell_{c+1} = c$**.]** Set $m = c$.
- [Step] We have an LFSR F_n with backtaps $\widetilde{T}_n$ of width ℓ_n generating $k_0, \ldots, k_{n-1}$ and an LFSR F_m generating $k_0, \ldots, k_{m-1}, \overline{k}_m$.
  - (a) If F_n generates $k_0, \ldots, k_{n-1}, k_n$ then set $\widetilde{T}_{n+1} = \widetilde{T}_n$, $\ell_{n+1} = \ell_n$, and so $F_{n+1} = F_n$. Keep m as it is.
  - (b) If F_n generates $k_0, \ldots, k_{n-1}, \overline{k}_n$, let $U = \{\widetilde{t} + n - m : \widetilde{t} \in \widetilde{T}_m\}$ and let $\widetilde{T}_{n+1} = \widetilde{T}_n \triangle (U \cup \{n - m\})$ as in Proposition 4.2. 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).

Note that we need $\max \widetilde{T}_{n+1} \leq \ell_{n+1}$ for the LFSR F_{n+1} to be well-defined. We prove this as part of Theorem 4.8.

Example 4.5. We apply the Berlekamp–Massey algorithm to the keystream $(1,1,1,0,1,0,1,0,0,0)$ from Example 4.3. After initialization we have $T_0 = \varnothing$, $\ell_0 = 0$, $T_1 = \{1\}$, $\ell_1 = 1$. 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	$\widetilde{T}_n$	ℓ_n	m	$\widetilde{T}_m$	$n-m$	U	$\widetilde{T}_{n+1}$	ℓ_{n+1}
1	$\varnothing$	1	0	$\varnothing$	1	$\varnothing$	$\{1\}$	1
3	$\{1\}$	1	0	$\varnothing$	3	$\varnothing$	$\{1,3\}$	3
6	$\{1,3\}$	3	3	$\{1\}$	3	$\{4\}$	$\{1,4\}$	4
7	$\{1,4\}$	4	6	$\{1,3\}$	1	$\{2,4\}$	$\{2\}$	4
8	$\{2\}$	4	6	$\{1,3\}$	2	$\{3,5\}$	$\{3,5\}$	5

Exercise. Run the algorithm starting with step 1, in which you should define $\widetilde{T}_2 = \{1\}$, and finishing with step 6, in which you should define $\widetilde{T}_7 = \{1,4\}$. Example 4.3 and Exercise 4.4 then do steps 7 and 8.

Berlekamp–Massey theorem. To prove that the LFSRs defined by running the Berlekamp–Massey algorithm have minimal possible width we need the following proposition.

Proposition 4.6. *Let $n \geq \ell$. If an LFSR F of width ℓ generates the keystream $(k_0, k_1, \ldots, k_{n-1}, b)$ of length $n+1$ then any LFSR F' generating the keystream $(k_0, k_1, \ldots, k_{n-1}, \overline{b})$ has width ℓ' where $\ell' \geq n+1-\ell$.*

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

$$k_s = \sum_{\widetilde{t}' \in \widetilde{T}'} k_{s-\widetilde{t}'} \quad \text{for } \ell' \leq s < n. \tag{†′}$$

By (†) for F in the case $s = n$ we have

$$b = \sum_{\widetilde{t} \in \widetilde{T}} k_{n-\widetilde{t}}.$$

Observe that $n - \widetilde{t} < n$ and, by our assumption, $n - \widetilde{t} \geq n - \ell \geq \ell'$. Therefore (†′) holds for each summand $k_{n-\widetilde{t}}$. Substituting we get

$$b = \sum_{\widetilde{t} \in \widetilde{T}} \sum_{\widetilde{t}' \in \widetilde{T}'} k_{n-\widetilde{t}-\widetilde{t}'} = \sum_{\widetilde{t}' \in T'} \sum_{\widetilde{t} \in T} k_{(n-\widetilde{t}')-\widetilde{t}} = \sum_{\widetilde{t}' \in \widetilde{T}'} k_{n-\widetilde{t}'} = \overline{b}$$

where we swapped the order of summation, then used (†), then (†′), then the assumption that the F' keystream ends with $\overline{b}$. Hence $b = \overline{b}$, a contradiction. □

Recall that step n of the Berlekamp–Massey algorithm returns an LFSR F_{n+1} with backtaps $\widetilde{T}_{n+1}$ and width ℓ_{n+1} generating $k_0, \ldots, k_{n-1}, k_n$.

Lemma 4.7. *With the notation above, if $\widetilde{t} \in \widetilde{T}_{n+1}$ then $\widetilde{t} \leq \ell_{n+1}$, and so F_{n+1} is well-defined.*

Proof. Since $\widetilde{T}_c = \widetilde{T}_{c+1} = \varnothing$, the lemma holds for c and $c+1$. We work by induction, supposing the lemma holds in the cases of m and n.

If, in step n, case (a) applies then $\ell_{n+1} = \ell_n$ and $\widetilde{T}_{n+1} = \widetilde{T}_n$ and by induction we have $\widetilde{t} \leq \ell_n$ for all $\widetilde{t} \in \widetilde{T}_n$, as required. Suppose case (b) applies. By definition of m,

$$\ell_m < \ell_{m+1} = \ldots = \ell_n.$$

The algorithm tells us to take $\ell_{m+1} = \max(\ell_m, m+1-\ell_m)$; since $\ell_{m+1} > \ell_m$ we have $\ell_{m+1} = m+1-\ell_m$. Therefore $n+1-\ell_n = n+1-\big(m+1-\ell_m\big) = n-m+\ell_m$ and

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

By the lemma for m we have

$$\widetilde{s} + n - m \leq \ell_m + (n-m)$$

for all $\widetilde{s} \in \widetilde{T}_m$. Since

$$\widetilde{T}_{n+1} = \widetilde{T}_n \triangle (\{\widetilde{s} + n - m : \widetilde{s} \in T_m\} \cup \{n-m\})$$

it now follows from the lemma for n that $\widetilde{t} \leq \max(\ell_n, n-m+\ell_m)$ for all $\widetilde{s} \in \widetilde{T}_{n+1}$. By $(\star)$, $\widetilde{t} \leq \ell_{n+1}$, as required. □

Theorem 4.8. *With the notation above, ℓ_{n+1} is the least width of any LFSR generating $k_0 \ldots k_{n-1}k_n$.*

Proof. By assumption the keystream begins $k_0 = k_1 = \ldots = k_{c-1} = 0$ and $k_{c+1} = 1$. The empty LFSR F_c generating $k_0 \ldots k_{c-1} = 0 \ldots 0$ has width 0, clearly the minimum possible. An LFSR generating $k_0 \ldots k_{c-1}k_c = 0 \ldots 01$ must begin with a non-zero key, so has width at least $c+1$. Since $\ell_{c+1} = c+1$, the LFSR F_{c+1} has minimum possible width.

Suppose an LFSR of width ℓ generates $k_0k_1 \ldots k_n$. Then F generates $k_0k_1 \ldots k_{n-1}$. By induction, F_n has minimum width ℓ_n. Therefore $\ell \geq \ell_n$. If $\ell_{n+1} = \ell_n$ we have $\ell \geq \ell_{n+1}$, as required. Therefore we may suppose that $\ell_{n+1} > \ell_n$. Hence case (b) applies, so F_n generates $k_0k_1 \ldots \bar{k}_n$, and $\ell_{n+1} = n+1-\ell_n$. By Proposition 4.6, $\ell \geq n+1-\ell_n$. Therefore $\ell \geq \ell_{n+1}$. □

5. The Discrete Fourier Transform

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

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

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*

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

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

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

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

and

$$\mathbf{P}[f(X) = g(X)] = \tfrac{1}{2}(1 + \mathrm{corr}(f,g)),$$
$$\mathbf{P}[f(X) \neq g(X)] = \tfrac{1}{2}(1 - \mathrm{corr}(f,g)).$$

For example, $\mathrm{corr}(f,g) = 1$ if and only if f and g are the same function, $\mathrm{corr}(f,g) = \frac{1}{2}$ if and only if $\mathbf{P}[f(X) = g(X)] = \frac{3}{4}$ and $\mathrm{corr}(f,g) = 0$ 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 Exercise 7.1 and Example 7.2) that linear functions are often weak cryptographically. So are functions that are highly correlated with linear functions. For consistency with the main course, we number positions from 0 below.

Given $T \subseteq \{0, \ldots, n-1\}$, 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, $L_{\{t\}}(x_0, \ldots, x_{n-1}) = x_t$ returns the entry in position t and $L_\varnothing(x) = 0$ is the zero function.

Lecture 10

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 $T \subseteq \{0, \ldots, n-1\}$. If $S, T \subseteq \{0, \ldots, n-1\}$ then*

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

Example 5.6. Let $\mathrm{maj} : \mathbb{F}_2^3 \to \mathbb{F}_2$ be the majority vote function defined by $\mathrm{maj}\big((x_0, x_1, x_2)\big) = 1$ if and only if at least two of x_0, x_1, x_2 are true. Then

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

Moreover

$$(-1)^{\mathrm{maj}} = \tfrac{1}{2}(-1)^{L_{\{0\}}} + \tfrac{1}{2}(-1)^{L_{\{1\}}} + \tfrac{1}{2}(-1)^{L_{\{2\}}} - \tfrac{1}{2}(-1)^{L_{\{0,1,2\}}}.$$

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*

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

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

(a) *The functions $(-1)^{L_T}$ for $T \subseteq \{0, \ldots, n-1\}$* **[Corrected from** $\{1, \ldots, n\}$**]** *are an orthonormal basis for the vector space of functions $\mathbb{F}_2^n \to \mathbb{R}$.* **Lecture 11**

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

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

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

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

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.

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 7.3 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 8.5. Recall that we write elements of $\mathbb{F}_2^8$ as pairs (v, w) with $(v, w) \in \mathbb{F}_2^4$. By definition, $e_k\big((v, w)\big) = (v', w')$ 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

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

By definition

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

where the third line follows because the summand for (v, w) is the same for all 2^6 pairs with the same w_1 and w_2. In $\sum_{w_1,w_2\in\mathbb{F}_2}(-1)^{(w_1+k_1)(w_2+k_2)}$, 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 $\frac{1}{2^2}\sum_{w_1,w_2\in\mathbb{F}_2}(-1)^{(w_1+k_1)(w_2+k_2)} = \frac{1}{2}$ and

$$\operatorname{corr}(L_{\{0\}} \circ e_k, L_{\{2\}}) = \tfrac{1}{2}(-1)^{k_0+k_6}$$

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. The mean should be close to $\frac{1}{2}(-1)^{k_0+k_6}$, and the sign then 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.

Extra: correlations for the m-quadratic stream cipher. In Example 7.5 of the main course we used the LFSR F of width 5 with taps $\{0,2\}$ and the LFSR F' of width 6 and taps $\{0,1,3,4\}$ to define a keystream $u_0u_1u_2\ldots$ by

$$u_s = k_sk'_s + k_{s-1}k'_{s-1} + \cdots + k_{s-(m-1)}k'_{s-(m-1)}$$

for each $s \geq m-1$. If $s < m-1$ we set $u_s = 0$. To compute the correlation between u_s and a bit k_s of the keystream for F, it is helpful to use this probabilistic interpretation of correlation.

Lemma 6.4. *Let $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ be Boolean functions. Let X be a random variable uniformly distributed on $\mathbb{F}_2^n$.*

$$\operatorname{corr}(f,g) = \mathbf{E}[(-1)^{f(X)+g(X)}].$$

Proof. By definition the expected value is $\sum_{x\in\mathbb{F}_2^n} \mathbf{P}[X = x](-1)^{f(x)+g(x)}$. Since $\mathbf{P}[X = x] = \frac{1}{2^n}$ for each $x \in \mathbb{F}_2^n$, this agrees with Definition 5.1. □

Write elements of $\mathbb{F}_2^{2m}$ as pairs (x,y). The relevant Boolean functions in this case are $f : \mathbb{F}_2^{2m} \to \mathbb{F}_2$, defined by

$$f\big((x,y)\big) = x_0y_0 + \cdots + x_{m-1}y_{m-1}$$

and the linear function $L_{\{m-1\}}(x,y) = x_{m-1}$. Let $(X,Y) \in \mathbb{F}_2^{2m}$ be a random variable distributed uniformly at random. By the lemma,

$$\begin{aligned}\operatorname{corr}(f, L_{(m-1)}) &= \mathbf{E}[(-1)^{f(X,Y)+L_{\{m-1\}}(X,Y)}] \\ &= \mathbf{E}[(-1)^{X_0Y_0+X_1Y_1+\cdots+X_{m-1}Y_{m-1}+X_{m-1}}] \\ &= \mathbf{E}[(-1)^{X_0Y_0}]\mathbf{E}[(-1)^{X_1Y_1}]\ldots\mathbf{E}[(-1)^{X_{m-1}Y_{m-1}+X_{m-1}}]\end{aligned}$$

where the final line follows because X_0 is independent of X_1, and so on. Since $\mathbf{P}[X_0Y_0 = 0] = \frac{3}{4}$ and $\mathbf{P}[X_0Y_0] = \frac{1}{4}$, we have $\mathbf{E}[(-1)^{X_0Y_0}] = \frac{1}{2}$. A similar argument shows that $\mathbf{E}[(-1)^{X_{m-1}Y_{m-1}+X_{m-1}}] = \frac{1}{2}$. Therefore the correlation is $\frac{1}{2^m}$, as claimed.

Extra: linear cryptanalysis of DES. In M. Matui, *The first experimental cryptanalysis of the Data Encryption Standard*, CRYPTO 1994: Advances in Cryptology CRYPTO '94, pages 1–11, Springer 1994, Matsui gives an attack on DES using linear cryptoanalysis on 2^{43} plaintext/ciphertext pairs (and then some further, less expensive work). Since the keylength is 56, this attack is subexhaustive. It is still close to the best known attack on DES.