MT362/462/5462 CRYPTOGRAPHY I

MARK WILDON

How to follow this course: These notes give the logical structure of the course. Many of you may prefer to learn the material from the slides on Moodle. These have all the content in the notes, and extra informal quizzes. Use the videos on Moodle as backup and if you need explanation of the quizzes on the slides, or the formal quizzes on Moodle.

These notes are based in part on notes written by Dr Siaw-Lynn Ng. I would very much appreciate being told of any corrections or possible improvements.

You are warmly encouraged to ask questions in the plenary, group work and Q&A sessions. Sessions marked 'face-to-face' will also be streamed online.

- Tuesday 1pm, **Plenary problem solving** (face-to-face) ARTS LT1,
- Wednesday 12 noon, **Group Work** (face-to-face), MFOX-SEM
- Friday 10am, **Q&A session** (online)
- Friday 3pm, **Group Work** (online)

I am also happy to answer questions about the lectures or problem sheets by email. My email address is `mark.wildon@rhul.ac.uk`.

MSc students doing MT5462: You have an extra session at Tuesday 11am ALT2. Also there are separate printed notes (like these), slides and videos on the Moodle page. These are marked M.Sc.

Office hour in McCrea LGF 0-25 or online: Thursday 2pm. The MS Teams link is `https://tinyurl.com/y38jovro`. It is also on the Moodle page.

Group work. Your timetable will show face-to-face (Wednesday 12 noon MFOX-SEM) and online (Friday 3pm) group work sessions in alternating weeks. Note that Teaching Week 1 is Week 2 of term.

- Session A
  - Face-to-face group work in Teaching Weeks: 1, 3, 5, 8, 10
  - Online group work in Teaching Weeks: 2, 4, 7, 9, 11
- Session B
  - Face-to-face group work in Teaching Weeks: 2, 4, 7, 9, 11
  - Online group work in Teaching Weeks: 1, 3, 5, 8, 10

If you intend to follow the course online only, you should still attend the scheduled face-to-face group work session on your timetable: online small groups will be formed so that you can take part.

Date: First Term 2020/21.

CIPHER SYSTEMS

We will study symmetric and public key ciphers and understand how they promise confidential communication. We will also see how they have been attacked, and in many cases defeated, using mathematical ideas from linear algebra, elementary number theory, probability theory, and statistics.

Outline.

(A) *Introduction:* alphabetic ciphers including the Vigenère cipher and one-time-pad. Statistical tests and applications of entropy. Security models and Kerckhoffs's Principle.
(B) *Stream ciphers:* linear feedback shift registers and pseudo-random number generation. Non-linear stream ciphers.
(C) *Block ciphers:* design principles, Feistel networks, DES and AES. Differential cryptanalysis.
(D) *Public key ciphers and digital signatures:* one-way functions, Diffie–Hellman, RSA and ElGamal. Factoring and discrete logs. Hash functions and signatures. Extra (and non-examinable): the Bitcoin blockchain.

The MT5462 course has additional material on boolean functions, the Berlekamp–Massey algorithm and linear cryptanalysis of block ciphers. Separate lecture notes will be issued.

Recommended Reading. All these books are in the library. If you find there are not enough copies, email me.

[1] *Cryptography, theory and practice*, D. Stinson, Chapman & Hall / CRC (2006). Concise and usually very clear, covers all the course (and more), 001.5436 STI (multiple copies, some on short loan).

[2] *Introduction to cryptography with coding theory*, W. Trappe and L. C. Washington, Pearson / Prentice Hall (2006), 001.5436 TRA. Similar to [1], but a bit more relaxed with more motivation.

[3] *Cryptography: a very short introduction*, F. C. Piper and S. Murphy, Oxford University Press (2002). A nice non-technical overview of cryptography: you can read it online via the library website.

[4] *Codes and cryptography*, D. Welsh, Oxford University Press (1988), 001.5436 WEL. Goes into more detail on some of the M.Sc. topics. Also useful for MT341/441/5441.

Prerequisites. You need basic probability, binary numbers and modular arithmetic. Probability is particularly important and conditional probability is reviewed in the videos on Moodle and quizzes.

Problem sheets. There will be 8 marked problem sheets; the first is due in on Friday 11th October. **15% of your final mark for the course is given for making a reasonable attempt at the problem sheets.**

Mathematica. The MATHEMATICA notebooks available from Moodle will be used for demonstrations in lectures and are useful (almost essential unless you want to do long encryptions by hand) for some problem sheet questions. You can get MATHEMATICA for free: search for 'Free Software Royal Holloway' or go to:

`https://intranet.royalholloway.ac.uk/students/help-support/it-services/it-essentials/free-software.aspx`

To get a notebook ready, select 'Evaluate Notebook' in the 'Evaluation' menu. If you find a freshly downloaded MATHEMATICA notebook gives errors, even after restarting MATHEMATICA, please email me at once.

Moodle. All handouts, problem sheets and answers will be posted on Moodle. Once you are registered for the course you should find a link under 'My courses'. If not please go to

`https://moodle.royalholloway.ac.uk/course/view.php?id=380`

This is the Moodle page for 462 (the M.Sci. course) and 5462 (the M.Sc. course) as well as 362: everyone should have access. If you find you do not, email me at once.

Exercises and quizzes. Exercises set in these notes are mostly simple tests that you are following the material. Some will be used for quizzes. There are also **many further quizzes** in the slides on Moodle, and some **formal quizzes on Moodle you must attempt, for 15% of your final grade**.

Optional questions and extras. Optional questions on problem sheets and any 'extras' in these notes are included for interest only, and to show you some mathematical ideas beyond the scope of this course. You should not worry if you find them difficult.

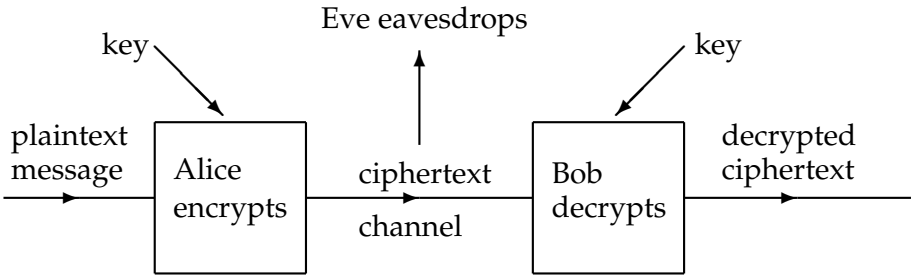
If you can do the compulsory questions on problem sheets, can do the quizzes on the slides and on Moodle, and can prove the results whose proofs are marked as examinable in these notes, then you should do very well in the examination.

(A) Introduction: alphabetic ciphers and the language of cryptography

1. Introduction: security and Kerckhoffs's Principle

This course is about the mathematics underlying cryptography. But you will not get the point unless you understand the overall objective.

As a basic model, Alice wants to send Bob a *plaintext* message. This message may be observed in the channel by the eavesdropper Eve, or intercepted and then modified by Malcolm, the Man-in-the-Middle. So Alice first encrypts the plaintext using some secret *key* known to her and Bob. At the other end Bob decrypts the *ciphertext*.



Here the key is the same for Alice and Bob. In Part D we look at public key cryptography, where Bob knows more about the key than Alice. But either way, it is *essential* that Bob knows something Eve/Malcolm do not.

Alice and Bob may have any of the following *security requirements*.

- **Confidentiality**: Eve cannot read the message.
- **Data integrity**: any change made by Malcolm to the ciphertext is detectable.
- **Authentication**: Alice and/or Bob are who they claim to be.
- **Non-repudiation**: Alice cannot plausibly deny she sent the message.

Example 1.1.

(0) For email the channel is the internet. Every email you send is typically received and sent on by multiple computers before it reaches its destination. Unless you arrange your own encryption, any rogue system-administrator can easily read your email.

(1) If you encrypt a file using a password on your computer, you require confidentiality and data integrity. In this case, you are Alice, and Bob is you a week later. The channel is the hard-disk (or SSD) in your computer.

(2) Using online banking to make a payment, both you and the bank require authentication and data integrity. The bank also requires non-repudiation. It is good practice to use two-factor authentication, so ideally the key is a code sent to your mobile phone, or generated by a 'PIN-sentry' device, in addition to a password. The channel is the internet.

Kerckhoffs's Principle. It is obviously important in cryptography to be very clear about what is public information and what is private. Kerckhoffs's Principle is that

'all the security in a cryptosystem lies in the key'.

Thus the attacker is assumed to know everything about the method that Alice uses to encrypt and Bob uses to decrypt. The only thing the attacker does not know is which specific key is used.

Example 1.2. On Friday, Alice will learn Bob's final year exam result x while Bob is out of the country. Alice, Bob and their trusted friend Trevor agree this method.

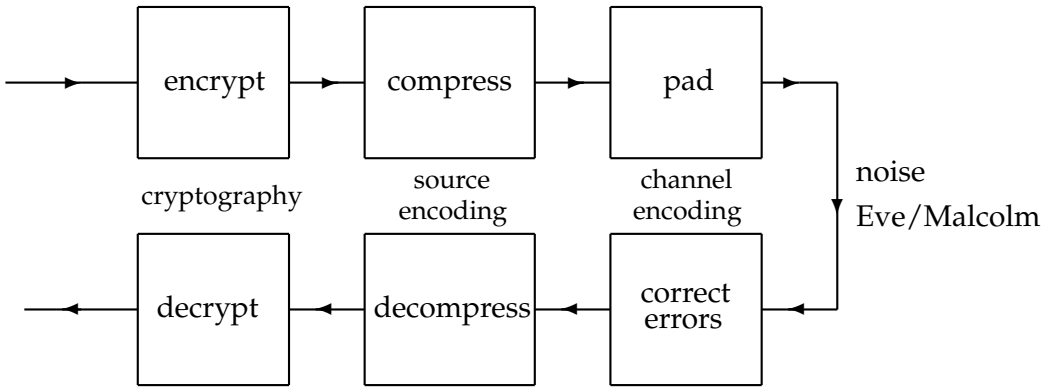
- On Monday, Trevor chooses a key $k \in \{0, 1, \ldots, 99\}$. He meets Alice and secretly tells her k. He meets Bob and secretly tells him k.
- On Tuesday, Bob leaves for Borneo. He can read email. Bob cannot send email or communicate in any other way.
- On Friday, Alice learns the plaintext $x \in \{0, 1, \ldots, 99\}$ and emails Bob the ciphertext $(x + k) \bmod 100$.

By Kerckhoffs's Principle, all this, except for the value of k, is known to the whole world. Eve, the eavesdropper, also learns y, the ciphertext sent by Alice to Bob.

(a) Can Eve learn anything about the plaintext x from the ciphertext y?
(b) What can Eve learn about the key from Alice's email? [*Hint:* Eve will certainly know that most marks are between 50 and 80.]
(c) Find some other problems in the scheme.
(d) Suppose that next year Alice sends Bob her own exam result $x' \in \{0, 1, \ldots, 99\}$ *using the same key*. What can Eve learn now?

The big picture. The extended diagram below shows how cryptography fits into the broader setting of communication theory. You can learn about source encoding (for compression) in MT341/441/5441 Channels and channel encoding (for error correction) in MT341/441/5441 Channels and MT361/461/5461 Error Correcting Codes, but there is no need to do these courses to understand this one.

In some cases, for example, sending an encrypted zip file, source encoding might be done before encryption, rather than after.



The mathematical model. As a mathematical model we suppose that there is a set $\mathcal{P}$ of *plaintexts*, a set $\mathcal{C}$ of *ciphertexts*, and a set $\mathcal{K}$ of *keys*. For each key $k \in \mathcal{K}$ there is an encryption function $e_k : \mathcal{P} \to \mathcal{C}$.

Note that mathematical functions are 'deterministic': if you put in plaintext x to the encryption function e_k, you will always get the same ciphertext out, namely $e_k(x)$. It is *only* by varying the key that the same plaintext can be encrypted as different ciphertexts.

Exercise 1.3. In Example 1.2, we have $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0, 1, \ldots, 99\}$.

(a) What are the encryption functions $e_k : \mathcal{P} \to \mathcal{C}$ in Example 1.2?
(b) Generally, what properties should the encryption functions have?

We answer (b) as part of the formal Definition 3.1 below of cryptosystems.

2. Alphabetic ciphers

Question. Suppose Eve, the eavesdropper, observes a ciphertext. What can she deduce about the plaintext and the key?

We shall answer this basic question for some ciphers that operate directly on English letters and words. It is a useful convention to write plaintexts in `lower case` and ciphertexts in `UPPER CASE`.

Caesar and substitution ciphers.

Example 2.1. The *Caesar cipher* with key $k \in \{0, 1, \ldots, 25\}$ encrypts a word by shifting each letter s positions forward in the alphabet, wrapping round at the end. For example if the key is 3 then '`hello`' becomes `KHOOR` and '`zany`' becomes `CDQB`. The table below shows all 26 possible shifts.

0	ABCDEFGHIJKLMNOPQRSTUVWXYZ	13	NOPQRSTUVWXYZABCDEFGHIJKLM
1	BCDEFGHIJKLMNOPQRSTUVWXYZA	14	OPQRSTUVWXYZABCDEFGHIJKLMN
2	CDEFGHIJKLMNOPQRSTUVWXYZAB	15	PQRSTUVWXYZABCDEFGHIJKLMNO
3	DEFGHIJKLMNOPQRSTUVWXYZABC	16	QRSTUVWXYZABCDEFGHIJKLMNOP
4	EFGHIJKLMNOPQRSTUVWXYZABCD	17	RSTUVWXYZABCDEFGHIJKLMNOPQ
5	FGHIJKLMNOPQRSTUVWXYZABCDE	18	STUVWXYZABCDEFGHIJKLMNOPQR
6	GHIJKLMNOPQRSTUVWXYZABCDEF	19	TUVWXYZABCDEFGHIJKLMNOPQRS
7	HIJKLMNOPQRSTUVWXYZABCDEFG	20	UVWXYZABCDEFGHIJKLMNOPQRST
8	IJKLMNOPQRSTUVWXYZABCDEFGH	21	VWXYZABCDEFGHIJKLMNOPQRSTU
9	JKLMNOPQRSTUVWXYZABCDEFGHI	22	WXYZABCDEFGHIJKLMNOPQRSTUV
10	KLMNOPQRSTUVWXYZABCDEFGHIJ	23	XYZABCDEFGHIJKLMNOPQRSTUVW
11	LMNOPQRSTUVWXYZABCDEFGHIJK	24	YZABCDEFGHIJKLMNOPQRSTUVWX
12	MNOPQRSTUVWXYZABCDEFGHIJKL	25	ZABCDEFGHIJKLMNOPQRSTUVWXY

Exercise 2.2.

(a) Mark (the mole) knows that the plaintext is 'apple' and the ciphertext is CRRNG. Show that Mark can deduce the key.

(b) Eve (the eavesdropper) has observed the ciphertext ACCB. What is the key and what is the plaintext?

(c) Suppose instead Eve observes GVTJPO. What can she deduce? Suppose Eve later observes BUPN. What does she conclude?

Barring the (very exceptional behaviour) in (c), the key can typically be deduced from a single ciphertext; (a) shows that the Caesar cipher can always be broken if both a plaintext and its ciphertext are known.

Example 2.3. Let $\pi : \{\mathtt{a}, \ldots, \mathtt{z}\} \to \{\mathtt{A}, \ldots, \mathtt{Z}\}$ be a bijection. The *substitution cipher* e_π applies π to each letter of a plaintext in turn. For example, if

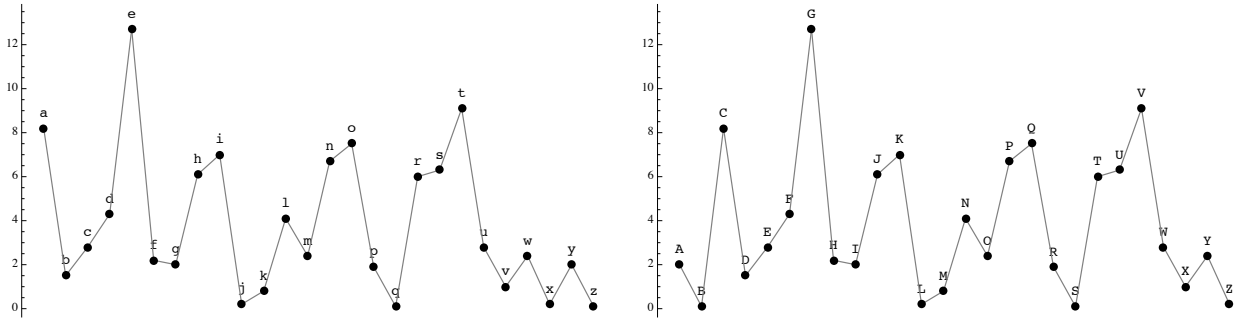
$$\pi(\mathtt{a}) = \mathtt{Z},\ \pi(\mathtt{b}) = \mathtt{Y},\ \ldots,\ \pi(\mathtt{z}) = \mathtt{A}$$

then $e_\pi(\texttt{hello there}) = \texttt{SVOOL GSVIV}$. (In practice spaces were deleted before encryption, but we will keep them to simplify the cryptanalysis.) The Caesar cipher with key k is the special case where π shifts each letter forward k times.

The table below (taken from Stinson's book [1]) shows the frequency distribution of English, most frequent letters first. Probabilities are given as percentages.

e	t	a	o	i	n	s	h	r	d	l	u	c
12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3	4.0	2.8	2.8
m	w	f	g	y	p	b	v	k	j	x	q	z
2.4	2.3	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.2	0.1	0.1	0.1

The frequency distribution of English is shown in the graph on the left below.



Using a substitution cipher, the probability distribution of ciphertext letters is a rearrangement of the probability distribution of plaintext letters. In particular, there are still three peaks, corresponding to e, t, and a.

For example, the graph on the right above shows the distribution in the (very) special case where π is the Caesar shift by 2, so $\pi(\mathtt{a}) = \mathtt{C}, \pi(\mathtt{b}) = \mathtt{D}, \ldots, \pi(\mathtt{z}) = \mathtt{B}$.

A sufficient long ciphertext can be decrypted by using frequency analysis to deduce $\pi(\mathtt{e}), \pi(\mathtt{t}), \ldots$, and then guessing likely words. For example, '`hello there`' has 'e' as its most common character. Correspondingly, when encrypted using the Caesar shift by 2, $\pi(\mathtt{e}) = \mathtt{G}$ is the most common character in the ciphertext.

It is also helpful to look for common digrams and trigrams such as '`th`', '`he`', '`in`', '`er`', '`the`', '`and`', '`ing`': see Example 2.5 and Example 2.7.

Exercise 2.4. How many substitution ciphers are there? [*Hint:* count the keys π by multiplying choices. There are 26 choices for $\pi(\mathtt{a})$, then 25 choices for $\pi(\mathtt{b})$, since we can choose any letter in $\{\mathtt{A}, \ldots, \mathtt{Z}\}$ *except* $\pi(\mathtt{a})$, and so on.]

Example 2.5. Eve observes the ciphertext

```
KQX WJZRUHXZKUY GTOXSKPIX GW SMBFKGVMUFQB PL KG XZUTYX KDG
FXGFYX JLJUYYB MXWXMMXR KG UL UYPSX UZR TGT KG SGHHJZPSUKX
GIXM UZ PZLXSJMX SQUZZXY PZ LJSQ U DUB KQUK UZ GFFGZXZK
XIX SUZZGK JZRXMLKUZR DQUK PL TXPZV LUPR KQX SQUZZXY SGJYR
TX U KXYXFQGZX YPZX GM KQX PZKXMZXK WGM XCUHFYX
```

The frequencies, again expressed as percentages, of all the letters are shown below. (All the donkey work in this example can be done using the MATHEMATICA notebook `AlphabetCiphers` available on Moodle.)

X	Z	U	K	G	Y	S	P	M	Q	L	J	F
14.7	10.3	9.5	8.6	7.7	5.2	4.7	4.7	4.7	4.3	3.4	3.4	3.4
R	T	W	H	B	I	D	V	O	C	N	E	A
3.0	2.6	1.7	1.7	1.7	1.3	1.3	0.9	0.4	0.4	0	0	0

The first word is `KQX`; this also appears in the final line, and `X` is comfortably the most common letter. We guess that `KQX` is 'the' and that `ZUKG` are most probably four of the letters 'taoin'. Since `U` appears on its own, it is probably 'a' or 'i', and from `KQUK` in line 3 it seems `U` is 'a'. Since the digraph `UZ` cannot be 'at', it is probably 'an'. Substituting for `KQXUZ` gives

```
the WJnRaHentaY GTOeStPIe GW SMBFtGVMaFhB PL tG enaTYe tDG
FeGFYe JLJaYYB MeWeMMeR tG aL aYPSe anR TGT tG SGHHJnPSate
GIeM an PnLeSJMe ShanneY Pn LJSh a DaB that an GFFGnent
eIe SannGt JnReMLtanR Dhat PL TePnV LaPR the ShanneY SGJYR
Te a teYeFhGne YPne GM the PnteMnet WGM eCaHFYe
```

From here it should not be too hard to decrypt the ciphertext. Good words to guess are '`teYeFhGne`' and '`PnteMnet`' in the bottom line and '`ShanneY`' in two lines above.

Exercise 2.6.

(a) After deciphering in Example 2.5, Eve knows that $\pi(\mathtt{a}) = \mathtt{U}$, $\pi(\mathtt{b}) = \mathtt{T}$, and so on. Does she know the key π?

(b) Will Eve have any difficulty in decrypting further messages encrypted using the same substitution cipher?

(c) Can Malcolm (the man-in-the-middle) successfully alter the ciphertext?

The substitution cipher is weak mainly because it is possible to start with a guess for the key, say π, that is partially correct, and then improve it step-by-step by looking at the decrypt $e_\pi^{-1}(y)$ implied by this key.

Example 2.7. To make this process automatic, we need a quantitative way to measure how 'close to English' $e_\pi^{-1}(y)$ is. Recall that a *trigram* is three consecutive letters. A good scoring function is $\sum_t \log p_t$ where the sum is over all trigrams t in $e_\pi^{-1}(y)$ and p_t is the probability of the trigram t in English. (This is motivated by maximum likelihood estimation.) For example, the three most common trigrams in English are '`the`', '`and`', '`ing`' with probabilities

$$p_{\mathtt{the}} = 0.01876,\ p_{\mathtt{and}} = 0.00750,\ p_{\mathtt{ing}} = 0.00742$$

and the score of '`there`' is

$$p_{\mathtt{the}} + p_{\mathtt{her}} + p_{\mathtt{ere}} = \log 0.01876 + \log 0.00370 + \log 0.00321 = -15.317.$$

Note the score is always negative, since $\log p_t < 0$ for each trigram t.

We start with the guess for the key given by frequency analysis. In each step we swap the encryptions of two plaintext letters. Since you can sort a deck of cards by repeatedly choosing two cards and then swapping them, this means all 26! (see Exercise 2.4) possible keys can be reached by taking enough steps. In Example 2.5 the guess from frequency analysis is

$$\pi(\mathtt{a}) = \mathtt{U}, \pi(\mathtt{b}) = \mathtt{B}, \pi(\mathtt{c}) = \mathtt{F}, \pi(\mathtt{d}) = \mathtt{Q}, \pi(\mathtt{e}) = \mathtt{X}, \pi(\mathtt{f}) = \mathtt{W}, \pi(\mathtt{g}) = \mathtt{H}\ldots,$$
$$\ldots, \pi(\mathtt{p}) = \mathtt{I}, \pi(\mathtt{q}) = \mathtt{E}, \ldots, \pi(\mathtt{y}) = \mathtt{D}, \pi(\mathtt{z}) = \mathtt{A}$$

with score -2868.97, implying that the plaintext is

```
ode futmayetoan iwkesohpe if srgcoivracdg hl oi etawne
obi ceicne uluanng referrem oi al anhse atm wiw oi
siyyuthsaoe iper at htlesure sdatten ht lusd a bag ...
```

The first step is chosen to maximize the increase in the score. Scoring using the 3000 most common trigrams, it turns out to be optimal to swap the encryptions of '`b`' and '`g`'. The new guess for the key is π' where

$\pi'(\texttt{b}) = \texttt{H}$ and $\pi'(\texttt{g}) = \texttt{B}$; otherwise π' agrees with π. The score improves to -2868.68.

After 40 steps the implied plaintext is

```
rie futhametral ockedryve of dngprownapig ys ro etacle
rbo people usuallg nefenneh ro as alyde ath coc ro
dommutydare oven at ytsedune diattel yt sudi a bag ...
```

with score -2465.68. Looking at `people` and `usuallg` we can see this gives a much better approximation to the key. After 53 steps the implied plaintext is entirely correct! Try it online at `repl.it/@mwildon/SubstitutionHillClimbWeb`.

Exercise 2.8. The strategy in Example 2.7 is called 'hill-climbing'. Why this name?

In the substitution cipher the same bijection is applied to every position in the plaintext. Choosing a different bijection for some positions, even using only Caesar shifts, gives a stronger cipher.

Vigenère cipher. We need some more mathematical notation. Define a bijection between the alphabet and $\{0, 1, \ldots, 25\}$ by

$$\texttt{a} \longleftrightarrow 0, \texttt{b} \longleftrightarrow 1, \ldots, \texttt{z} \longleftrightarrow 25.$$

Using this bijection we identify a word of length ℓ with an element of $\{0, 1, \ldots, 25\}^\ell$. For example, '$\texttt{hello}$' $\longleftrightarrow (7, 4, 11, 11, 14) \in \{0, 1, \ldots, 25\}^5$.

After converting letters to numbers, the Caesar cipher with shift s becomes the function $c \mapsto c + s \bmod 26$.

Definition 2.9. The key k for the *Vigenère cipher* is a string. Suppose that k has length ℓ. Given a plaintext x with its spaces deleted, we define its encryption by

$$e_k(x) = (x_0 + k_0, x_1 + k_1, \ldots, x_{\ell-1} + k_{\ell-1}, x_\ell + k_0, x_{\ell+1} + k_1, \ldots)$$

where $x_i + k_i$ is computed by converting x_i and k_i to numbers and adding them mod 26.

Note that after ℓ letters we 'wrap around', by re-using position 0 of the key, so $y_\ell = x_\ell + k_0 \bmod 26$, $y_{\ell+1} = x_{\ell+1} + k_1 \bmod 26$, and so on.

As seen in this definition, in this course we number positions in tuples from 0. This often works best in cryptography: for example, it means that $e_k(x)_i = x_i + k_{i \bmod \ell}$ for all $i \in \mathbb{N}_0$. (Numbering from 1, the case when $i = \ell$ would have to be defined separately.)

Example 2.10. Take $k =$ `bead`, so k has length 4. Under the bijection between letters and numbers, `bead` $\longleftrightarrow (1, 4, 0, 3)$. The table below shows that

$$e_{\texttt{bead}}(\texttt{meetatmidnightnear}) = \texttt{NIEWBXMLERIJIXNHBV}.$$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x_i	m	e	e	t	a	t	m	i	d	n	i	g	h	t	n	e	a	r
	12	4	4	19	0	19	12	8	3	13	8	6	7	19	13	4	0	17
k_i	b	e	a	d	b	e	a	d	b	e	a	d	b	e	a	d	b	e
	1	4	0	3	1	4	0	3	1	4	0	3	1	4	0	3	1	4
$x_i + k_i$	13	8	4	22	1	23	12	11	4	17	8	9	8	23	13	7	1	21
	N	I	E	W	B	X	M	L	E	R	I	J	I	X	N	H	B	V

Exercise 2.11.

(i) If you had to guess, which of the following sequences of 50 characters would you say was more likely to be a sample of letters (not necessarily adjacent) from a Caesar cipher ciphertext?

`UWBBJSNMXUBSOWGFZTUIFFBIIJUBSTBUNGFIBSJETSGMJPTOOB`
`UIWRBKBDJTSONEMOXSULBTSNOEWLGEFAZATEUIINFBFIBEIHID`
`ULIVWIRBBAKZBVDKJWTRSCOINVEOMMOWXESVUMLOBJTHSENLOX`

To help you decide, the table below shows the frequencies of the ten most common letters, and the rest all lumped together.

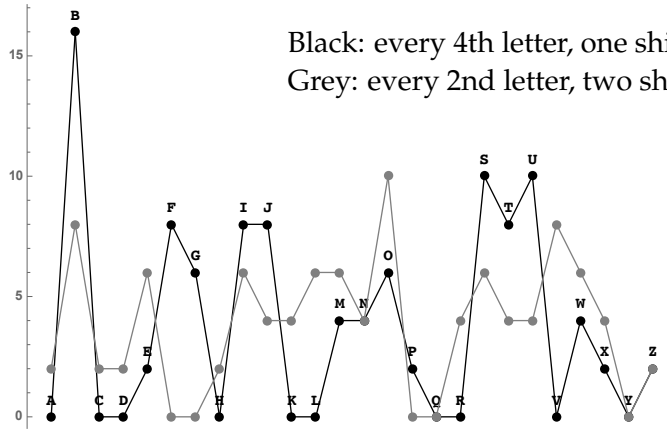
(A)	B	U	S	T	J	I	F	O	G	W	the rest
	8	5	5	4	4	4	4	3	3	2	8
(B)	D	V	L	K	C	Y	F	Z	U	R	the rest
	6	5	5	3	3	3	3	3	3	2	14
(C)	L	X	D	C	Y	M	K	I	F	B	the rest
	5	4	4	3	3	3	3	3	3	2	17

(ii) Suppose we take every kth character from a Vigenère ciphertext. Why will it have the most 'spiky' frequency distribution when k is the length of the key?

The samples in (i) are 50 every 4th, every 2nd and every from the ciphertext y in Example 2.16 below, encrypted using the Vigenère cipher with the key `bead`, *and taking just 50 letters in total*.

When we take every 4th letter of y, we get the plaintext shifted forwards by 1 (since $b \longleftrightarrow 1$) and the probability distribution is the shift by 1 of the English distribution shown on page 7. For instance, the peaks at `B` and `S` correspond to `a` and `t`. (The sample is small; while `e` was not the most common letter, there is still a high value at `F`.)

When we take every letter of y we get the average of four shifts of the English distribution, making the distribution much closer to uniform. This can be seen by comparing the black and grey lines in the graph below, produced using `AlphabetCiphers.nb`



To make this idea of 'spikiness' and 'uniformity' precise, we need to measure it by a statistic we can compute given a ciphertext.

Definition 2.12. The *Index of Coincidence* of a ciphertext y, denoted $I(y)$, is the probability that two entries of y, chosen at random from different positions, are equal.

Exercise 2.13. Explain why $I(\texttt{QXNURA}) = I(\texttt{QNRFLX}) = 0$ and check that $I(\texttt{MOODLE}) = \frac{1}{15}$. What is $I(\texttt{AAABBC})$?

There is a simple formula for $I(y)$. (An examinable proof: see the slides and videos!)

Lemma 2.14. *If the ciphertext y of length n has exactly f_i letters corresponding to i, for each $i \in \{0, 1, \ldots, 25\}$ then*

$$I(y) = \sum_{i=0}^{25} \frac{f_i(f_i - 1)}{n(n-1)}.$$

Attack 2.15. *Given a Vigenère ciphertext y, take every kth letter for all small k. For instance when $k = 3$ the sample is $y_0y_3y_6y_9\ldots$ and when $k = 4$ the sample is $y_0y_4y_8\ldots$. The Index of Coincidence will be greatest (for long samples) when we split at the key length, ℓ. Now $y_0y_\ell y_{2\ell}\ldots$ have all been encrypted by shifting by k_0: assuming that the most common letter is the shift of '*`e`*' determines the shift. Repeat with $y_1y_{\ell+1}y_{2\ell+1}\ldots$ to determine k_1, and so on, up to $k_{\ell-1}$.*

The Index of Coincidence of English is about 0.066. You don't need to memorize this: in practice you can just pick the highest value!

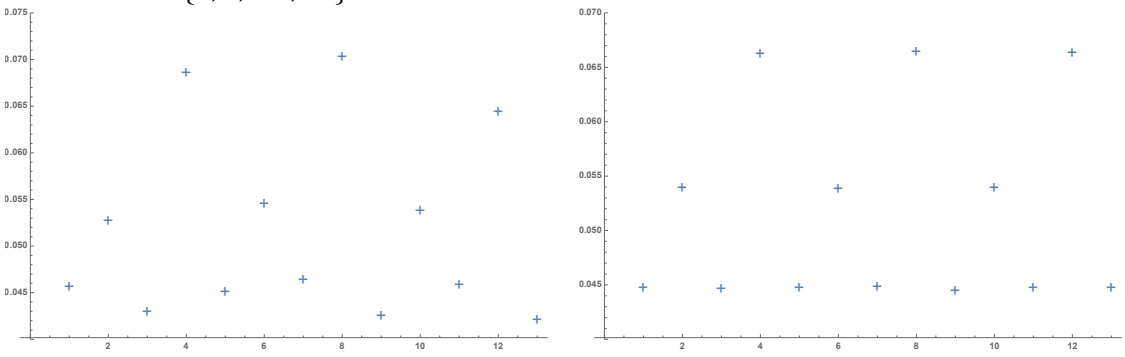
Example 2.16. The final part of Chapter 1 of *Persuasion* by Jane Austen begin

> This very awkward history of Mr Elliot was still, after an interval of several years, felt with anger by Elizabeth, who had liked the man for himself, and still more for being her father's heir, and whose strong family pride could see only in him a proper match for Sir Walter Elliot's eldest daughter. ....

After deleting spaces and punctuation and encrypting using the Vigenère cipher with key 'bead', the ciphertext is

ULIVWIRBBAKZBVDKJWTRSCOINVEOMMOWXESVUMLOBJTHSENL ...

The graph on the left below shows the mean Index of Coincidence when the ciphertext is split taking every kth letter, starting at the first, for $k \in \{1, 2, \ldots, 13\}$.



While there is a slight decrease from 8 to 12, the similar values for 4 and 8 suggest the key has length 4. This is confirmed by splitting the ciphertext taking every kth letter, starting at each of the initial k letters in turn, and then taking the mean of k different IOCs. This bigger sample (improving on Attack 2.15 as described above), gives the more accurate picture shown in the graph on the right above

Continuing with the attack, we now take every four letter of the ciphertext starting at the first (in position 0) to get

$$y_0 y_4 y_8 \ldots = \text{'UWBBJSNMXUBSOWGFZTUIFFBIIJUB}\ldots\text{'}$$

These ciphertext letters have all been shifted in the same way. The percentage frequency table (as in Example 2.5) begins

F	P	U	O
12.5	8.3	8.0	8.0

Assuming 'F' $\longleftrightarrow$ 5 is the encryption of 'e' $\longleftrightarrow$ 4, the shift in the Caesar cipher is 1 $\longleftrightarrow$ 'b', so we correctly guess the first letter of the key is 'b'. The MATHEMATICA notebook on Moodle shows this simple strategy works in all 4 positions to reveal the key bead.

Exercise 2.17. Why are there are smaller peaks at 2, 6 and 10 in the plot of Indices of Coincidence above? [*Hint:* taking every 2nd position only two of the four shifts are seen. The same is true taking every 6th . . .].

Extra: A more reliable method. Suppose you know the key length in the Vigenère cipher is ℓ and you have a sample $y_p y_{\ell+p} y_{2\ell+p} \ldots$, obtained by taking every ℓth letter in the plaintext x, starting at position p. Let x be an English sample of about the same length as $y_p y_{\ell+p} y_{2\ell+p} \ldots$. A better way to find the Caesar shift for position p is to concatenate x with the text obtained by shifting each letter of $y_p y_{p+\ell} y_{p+2\ell} \ldots$ forward by s, for each $s \in \{0, 1, \ldots, 25\}$. The Index of Coincidence will be maximized when s is the Caesar shift, since then both parts of the sample are in English.

The f_i^2 in the numerator of the formula in Lemma 2.14 may remind you slightly of the χ^2-test.

Exercise 2.18. What statistic would you compute to do a χ^2-test that the distribution of the ciphertext is uniform?

Statistics can appear a dry subject. I hope this example has shown you that it can be both useful and interesting. For further examples, one only has to look at the many triumphs of machine learning (the buzzword for statistical inference), from 'Intelligent personal assistants' such as Siri to the shocking defeat of the world Go champion by AlphaGo.

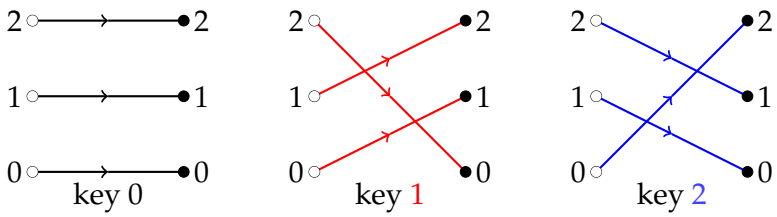
Extra: Different languages. The Index of Coincidence of English, estimated from a large sample, is about 0.066, and for German it is about 0.071. Since written German has 30 characters, so more ways for two characters to be different, it is fairer to compare $26 \times 0.066 \approx 1.72$ and $30 \times 0.068 \approx 2.14$. The most frequent letter in German is '`e`', at 16%, followed by '`n`' at 9.8% and then '`s`' at 7.2%. These all show that the frequency distribution of German is significantly 'spikier' than English. (The 'extra' German characters not seen in English are `ü`, `ö`, `ä`, `ß`.)

Exercise 2.19. For which language are alphabetic ciphers easier to break? (And, for amateur philologists, why is not surprising that German is spikier than English?)

3. Cryptosystems and perfect secrecy

Question. What, mathematically, is a cryptosystem? When can we be sure that Eve learns nothing about the plaintext from an observed ciphertext?

Cryptosystems. The three different encryption functions for the Caesar cipher with 'alphabet' $\{0, 1, 2\}$ are shown in the diagram below.



Definition 3.1. Let $\mathcal{K}, \mathcal{P}, \mathcal{C}$ be non-empty finite sets. A *cryptosystem* is a family of *encryption functions* $e_k : \mathcal{P} \to \mathcal{C}$ and *decryption functions* $d_k : \mathcal{C} \to \mathcal{P}$, one for each $k \in \mathcal{K}$, such that for each $k \in K$,

$$(\star) \qquad d_k(e_k(x)) = x$$

for all $x \in \mathcal{P}$. We call $\mathcal{K}$ the *keyspace*, $\mathcal{P}$ the set of *plaintexts*, and $\mathcal{C}$ the set of *ciphertexts*.

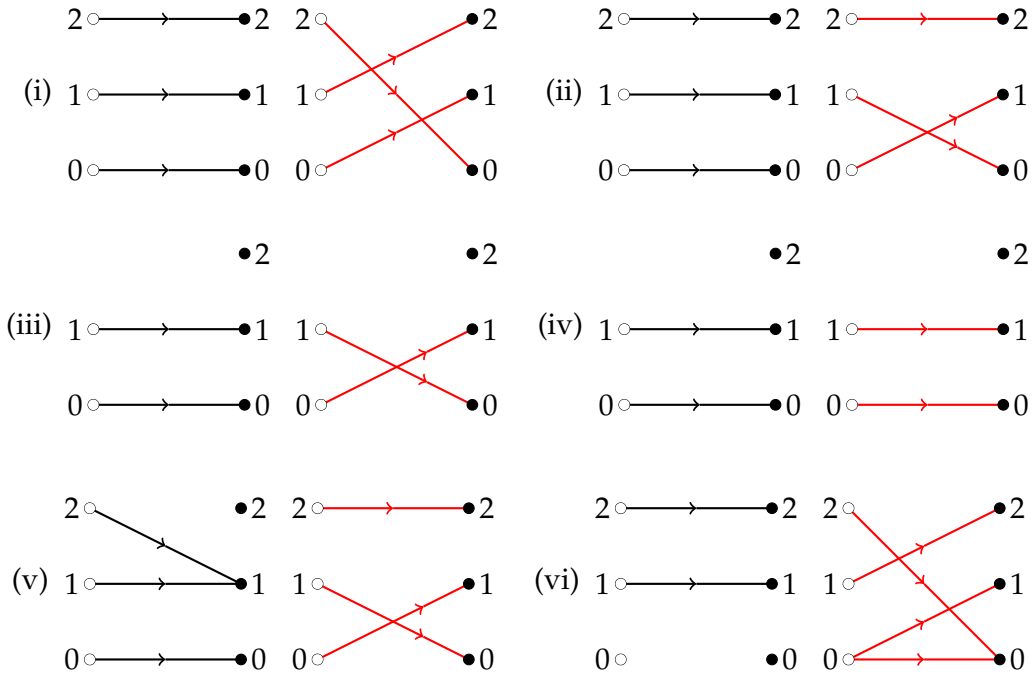
A function $f : \mathcal{P} \to \mathcal{C}$ is injective if for all $x, x' \in \mathcal{P}$, $f(x) = f(x')$ implies $x = x'$. Equivalently (contrapositive), if $x \neq x'$ then $f(x) \neq f(x')$.

Exercise 3.2.

(i) Use $(\star)$ to show that the encryption functions e_k in a cryptosystem are injective for all $k \in \mathcal{K}$.

(ii) Why do we want the encryption functions in a cryptosystem to be injective?

Exercise 3.3. Each diagram (i)–(vi) below each show two functions. Which are the encryption functions in a cryptosystem with two keys, one black and one red? Equivalently, when can decryption functions satisfying $(\star)$ be defined? In each case $\mathcal{P}$ is on the left-hand side and $\mathcal{C} = \{0, 1, 2\}$ is on the right-hand side.



In (v), the black encryption function is not injective. Correspondingly, someone using the cryptosystem with the black key does not know how to decrypt 1. This shows why we require ($\star$). In (vi) neither function is well-defined.

It may seem strange that (iv) is a cryptosystem: in practice it would be unusual for two keys to define the same encryption function. However checking that this is definitely *not the case* would be non-trivial for some practical ciphers, so we do not rule it out in the definition.

Exercise 3.4.

(i) An undergraduate writes 'For each $x \in \mathcal{P}$ there is a unique $y \in \mathcal{C}'$. Does this mean that e_k is injective?
(ii) Show that if $|\mathcal{P}| = |\mathcal{C}|$ then the encryption functions are bijections and $d_k = e_k^{-1}$ for each $k \in \mathcal{K}$.
(iii) Is there a cryptosystem with $|\mathcal{C}| < |\mathcal{P}|$?

Recall that $\mathbb{Z}_n$ denotes the set $\{0, 1, \ldots, n-1\}$ with addition and multiplication defined modulo n. (If you prefer the definition as a quotient ring, please feel free to use it instead.) For example $7 + 8 \equiv 4 \bmod 11$ and $7 \times 8 \equiv 1 \bmod 11$.

Example 3.5 (Numeric one-time pad)**.** Fix $n \in \mathbb{N}$. The *numeric one-time pad* on $\{0, 1, \ldots, n-1\}$ has $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_n$. The encryption functions are $e_k(x) = (x + k) \bmod n$. As expected from Exercise 3.4(ii), each e_k is a bijection, and the decryption functions are $d_k = e_k^{-1}$. Explicitly, $d_k(y) = (y - k) \bmod n$.

The diagrams before Definition 3.1 show the numeric one-time pad on $\{0, 1, 2\}$.

In Example 1.2 and Sheet 1 Question 2, Alice and Bob use the numeric one-time pad with $n = 100$. The key was given to them by their trusted friend Trevor, who was equally likely to pick each key.

- Suppose that Eve observes the ciphertext 80.
- The plaintext is x if and only if the key is $(80 - x) \bmod 100$.
- Since each key is equally likely then it seems reasonable to say that Eve learns nothing about the plaintext.

Moreover, as seen in the Group Work for Week 1, since the ciphertext is $x + k \bmod 100$, and all keys are equally likely, so are all ciphertexts.

Probability model. To make precise the idea that Eve learns nothing about the plaintext from an observed ciphertext we use a probability model. There are notes on Moodle reviewing basic probability theory.

Fix a cryptosystem in our usual notation. We make $\mathcal{K} \times \mathcal{P} \times \mathcal{C}$ a probability space by assuming that the plaintext $x \in \mathcal{P}$ is chosen *independently*

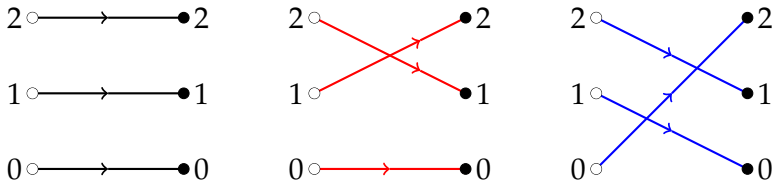
of the key $k \in \mathcal{K}$; the ciphertext is then $e_k(x)$. Thus if p_x is the probability the plaintext is $x \in \mathcal{P}$ and r_k is the probability the key is k then the probability measure is defined by

$$p_{(k,x,y)} = \begin{cases} r_k p_x & \text{if } y = e_k(x) \\ 0 & \text{otherwise.} \end{cases}$$

Let K, X, Y be the random variables standing for the key, plaintext and ciphertext, respectively.[1]

Exercise 3.6. Is the assumption that the key and plaintext are independent reasonable?

Example 3.7. The cryptosystem below will be used for a quiz in lectures.



Note the critical calculation using conditional probability (or Bayes' Theorem if you prefer):

$$\mathbf{P}[X = x|Y = y] = \frac{\mathbf{P}[Y = y|X = x]\mathbf{P}[X = x]}{\mathbf{P}[Y = y]}.$$

Here, and in general, $\mathbf{P}[Y = y|X = x] = \sum_{k \in \mathcal{K} : e_k(x) = y} \mathbf{P}[K = k]$ is a sum of key probabilities: we saw $\mathbf{P}[Y = 0|X = 0] = r_{\text{black}} + r_{\text{red}}$.

Suppose that each key is used with equal probability. Then $\mathbf{P}[Y = 0] = \mathbf{P}[Y = 0|X = 0]p_0 + \mathbf{P}[Y = 0|X = 0]p_1 + \mathbf{P}[Y = 0|X = 2]p_2 = \frac{2}{3}p_0 + \frac{1}{3}p_1$ and by the basic calculation

$$\mathbf{P}[X = 0|Y = 0] = \frac{\mathbf{P}[Y = 0|X = 0]p_0}{\mathbf{P}[Y = 0]} = \frac{\frac{2}{3}p_0}{\frac{2}{3}p_0 + \frac{1}{3}p_1 + 0p_2} = \frac{2p_0}{2p_0 + p_1}.$$

Whenever $p_2 > 0$ this probability is more than p_0, so Eve should now believe that plaintext 0 is more probable. In the language of Bayesian statistics, Eve's posterior probabilities are different to her prior probabilities.

Bluffers' guide:

- if you see $\mathbf{P}[Y = y|X = x]$ think '**nice**: key probability';
- if you see $\mathbf{P}[X = x|Y = y]$ think '**nasty**, turn it around'

[1] To be very formal, K, X and Y are the functions defined on the probability space $\mathcal{K} \times \mathcal{P} \times \mathcal{C}$ by $K(k, x, y) = k$, $X(k, x, y) = x$ and $Y(k, x, y) = y$.

Example 3.8. Consider the numeric one-time pad in Example 3.5, Assume that keys are chosen with equal probability $\frac{1}{n}$. Suppose that Eve observes the ciphertext y.

(a) By Question 1 on Problem Sheet 2, $\mathbf{P}[X = x|Y = y] = p_x$ for all $x, y \in \mathbb{Z}_n$. This is a precise statement that Eve learns nothing about the plaintext from observing y. (In the sense of Definition 3.11, the one-time pad has perfect secrecy.)

(b) Since $\mathbf{P}[K = k|Y = y] = \mathbf{P}[X = y - k|Y = y]$, (a) implies that

$$\mathbf{P}[K = k|Y = y] = p_{y-k}.$$

Thus the probability distribution $\mathbf{P}[K = k|Y = y]$ for k varying is a reflected shift of the probability distribution $\mathbf{P}[X = x]$ on plaintexts. Unavoidably, Eve learns something about the key. This was seen in the Group Work for Week 1 (video and written answers are available on Moodle). If however each plaintext is equally likely, then Eve learns nothing about the key. By (a), in either case, she learns nothing about the plaintext,

Shannon's Perfect Secrecy Theorem. In practice, the user of a cryptosystem needs to know how to choose the keys.

Definition 3.9. We define a *practical cryptosystem* to be a cryptosystem together with a probability distribution on the keys such that

(1) $\mathbf{P}[K = k] > 0$ for all $k \in \mathcal{K}$
(2) for all $y \in \mathcal{C}$ there exists $x \in \mathcal{P}$ and $k \in \mathcal{K}$ such that $e_k(x) = y$.

Exercise 3.10.

(a) Why are the two conditions in Definition 3.9 reasonable?
(b) Show that in a practical cryptosystem in which every plaintext may sent, $\mathbf{P}[Y = y] > 0$ for all $y \in \mathcal{C}$.

Unlike the definition of perfect secrecy, which goes back to Shannon's 1949 paper (see the extras for this part), Definition 3.9 is not a standard definition in cryptography. You will be reminded of it when it is required.

Definition 3.11. Fix a practical cryptosystem.

(i) Let p_x for $x \in X$ be a probability distribution on the plaintexts The cryptosystem has *perfect secrecy for the distribution* p_x if

$$\mathbf{P}[X = x|Y = y] = p_x$$

for all $x \in \mathcal{P}$ and all $y \in \mathcal{C}$ such that $\mathbf{P}[Y = y] > 0$.

(ii) The cryptosystem has *perfect secrecy* if it has perfect secrecy for every probability distribution on the plaintexts.

Some textbook accounts of Shannon's Theorem have minor errors from overlooking that $\mathbf{P}[Y = y] > 0$ is needed for the conditional probability in (i) to make sense. By Exercise 3.10(b), in a practical cryptosystem in which every plaintext may be sent, this condition always holds.

By Example 3.8(a) the one-time pad on $\mathbb{Z}_n$ has perfect secrecy when keys are used with equal probability. In Example 3.7 the cryptosystem does not have perfect secrecy when keys are used with equal probability: we saw that if $p_0 = 0$, $p_1 = 1 - q$ and $p_2 = q$ then $\mathbf{P}[X = 2|Y = 1] = 2q/(1 + q)$ which is equal to q if and only if $q = 0$ or $q = 1$.

The aim of the remainder of this section is to prove a theorem describing practical cryptosystems with perfect secrecy.

Theorem 3.12 (Shannon 1949)**.** *If a practical cryptosystem has perfect secrecy then*

(a) *For all $x \in \mathcal{P}$ and $y \in \mathcal{C}$ the events $X = x$ and $Y = y$ are independent and $\mathbf{P}[Y = y|X = x] = \mathbf{P}[Y = y] > 0$.*

(b) *For all $x \in \mathcal{P}$ and all $y \in \mathcal{C}$ there exists a key k such that $e_k(x) = y$.*

(c) $|\mathcal{K}| \geq |\mathcal{C}|$.

(d) *Suppose $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}| = n$. For all $x \in \mathcal{P}$ and all $y \in \mathcal{C}$ there exists a unique key $k \in \mathcal{K}$ such that $e_k(x) = y$. Moreover each key is used with equal probability $1/n$ and $\mathbf{P}[Y = y] = 1/n$ for all $y \in \mathcal{C}$.*

Exercise 3.13. How does the conclusion (b) in Theorem 3.12 differ from property (2) in the definition of a practical cryptosystem?

We will go through the proof below of Shannon's Theorem in the plenary session in Teaching Week 3. The slides break it down further with informal quiz questions: you are encouraged to stop reading here, and instead try the slides! The proofs are the same.

Proof of Theorem 3.12 (An examinable proof.) Fix a probability distribution on the plaintexts such that $p_x > 0$ for all $x \in \mathcal{P}$.

Proof of (a). Since the cryptoscheme has perfect secrecy, $\mathbf{P}[X = x|Y = y] = p_x$ for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$. (The event $Y = y$ that we condition on has positive probability by Exercise 3.10(b).) Hence $\mathbf{P}[X = x \text{ and } Y = y] = \mathbf{P}[X = x|Y = y]\mathbf{P}[Y = y] = p_x\mathbf{P}[Y = y]$. This shows that the events $X = x$ and $Y = y$ are independent and $\mathbf{P}[Y = y|X = x] = \mathbf{P}[Y = y] > 0$.

Proof of (b). For $x \in \mathcal{P}$ and $y \in \mathcal{C}$, let

$$\mathcal{E}_{xy} = \{k \in \mathcal{K} : e_k(x) = y\}.$$

Note that $\mathbf{P}[Y = y|X = x] = \mathbf{P}[k \in \mathcal{E}_{xy}]$. By (a) $\mathbf{P}[k \in \mathcal{E}_{xy}] > 0$ for all $x \in \mathbf{P}$, $y \in \mathcal{C}$. Hence each $\mathcal{E}_{xy}$ is non-empty.

Proof of (c). Fix $x \in \mathcal{P}$. Observe that $\mathcal{K} = \bigcup_{y \in \mathcal{C}} \mathcal{E}_{xy}$ where the union is disjoint. By (b) each $\mathcal{E}_{xy}$ is non-empty. Therefore

$$(\dagger) \qquad |\mathcal{K}| \geq \sum_{y \in \mathcal{C}} |\mathcal{E}_{xy}| \geq \sum_{y \in \mathcal{C}} 1 = |\mathcal{C}|$$

as required.

Proof of (d). By hypothesis $|\mathcal{K}| = |\mathcal{C}|$. Hence each inequality in (†) is an equality, and so $|\mathcal{E}_{xy}| = 1$ for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$. Equivalently, for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$, there exists a unique key (it is the unique key in $\mathcal{E}_{xy}$) such that $e_k(x) = y$. Fix $y^\star \in \mathcal{C}$ and, for each $x^\star \in \mathcal{P}$, let $k_{x^\star}$ be the unique key such that $e_{k_{x^\star}}(x) = y^\star$. If $k_x = k_{x'} = k$ then

$$e_k(x) = e_{k_x}(x) = y^\star = e_{k_{x'}}(x') = e_k(x').$$

But e_k is injective, hence $x = x'$. Therefore the keys k_x for $x \in \mathcal{P}$ are distinct, and since $|\mathcal{P}| = |\mathcal{K}|$ by hypothesis, these are all the keys. By (a),

$$\mathbf{P}[k = k_x] = \mathbf{P}[k \in \mathcal{E}_{xy^\star}] = \mathbf{P}[Y = y^\star | X = x] = \mathbf{P}[Y = y^\star]$$

is independent of $x \in \mathcal{P}$. Therefore each key is used with equal probability $1/n$ and by the equation above, $\mathbf{P}[Y = y^\star] = 1/n$. Since we could have chosen any $y^\star \in C$, this shows $\mathbf{P}[Y = y] = 1/n$ for all $y \in \mathcal{C}$. □

Some good questions to ask about a theorem, or a proof of a theorem, are 'What examples of it have I seen?', 'Did we use all the hypotheses?', 'Does the converse hold?'. These are explored on Problem Sheet 2. In particular, the optional Question 7(b) asks you to show the converse result stated below.

Proposition 3.14 (Converse to Theorem 3.12(d))**.** *Suppose that $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$, that each key is used with equal probability, and for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$, there exists a unique $k \in \mathcal{K}$ such that $e_k(x) = y$. Then the cryptosystem has perfect secrecy and each ciphertext is equally likely.*

In Example 3.5 (also seen in the Group Work for Week 1), we saw a special case of this proposition. As an *exercise*, check that the hypothesis of this proposition hold in this example. Rather than prove it in the limited 'live time' for the course, we will instead use the Week 3 Group Work to explore what it means in practice.

Example 3.15. Consider a practical cryptosystem with perfect secrecy in which $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0, 1, \ldots, n-1\}$. By (d) in Theorem 3.12 all the keys have equal probability. Moreover, for each $x, y \in \{0, 1, \ldots, n-1\}$ there exists a unique $k \in \{0, 1, \ldots, n-1\}$. such that $e_k(x) = y$. Therefore the cryptosystem is determined by the $n \times n$ matrix M where

$$M_{xy} = k \iff e_k(x) = y.$$

For example, the numeric one-time pad on $\{0,1,2\}$ has matrix

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

Note that rows/columns are numbered starting from 0 at the top/left. Conversely, by Proposition 3.14, given a $n \times n$ matrix in which every row and column has entries $\{0,1,\ldots,n-1\}$ there is a corresponding cryptosystem with perfect secrecy. Such matrices are called *Latin squares* and often arise in cryptography and coding theory.

Finally Question 8 on Problem Sheet 2 asks you to explore another natural question. (Answers will be posted after the deadline.)

Exercise 3.16. Is there a practical cryptosystem with perfect secrecy in which the ciphertexts have different probabilities?

4. Attack models

Exercise 4.1. Suppose Eve observes a ciphertext. Why is it more valuable to her to learn the key than the plaintext?

> **Question.** What can an attacker learn about the plaintext and key from an observed ciphertext? Can the key still be unknown when an attacker knows both a plaintext *and* its ciphertext?

Example 4.2 (Affine cipher)**.** Let q be prime. Let $\mathbb{Z}_q = \{0,1,\ldots,q-1\}$. The *affine cipher* on $\mathbb{Z}_q$ has $\mathcal{P} = \mathcal{C} = \mathbb{Z}_q$ and

$$\mathcal{K} = \{(a,c) : a \in \mathbb{Z}_q, c \in \mathbb{Z}_q, a \neq 0\}.$$

The encryption functions are defined by $e_{(a,c)}(x) = ax + c \bmod q$. The decryption functions are defined by $d_{(a,c)}(y) = b(y-c) \bmod q$, where $b \in \mathbb{Z}_q$ is the unique element such that $ab \equiv 1 \bmod q$. With these definitions, the affine cipher is a cryptosystem.

For example, in the affine cipher on $\mathbb{Z}_{11}$, $e_{(9,2)}(5) = 3$ since $9 \times 5 + 2 \equiv 3 \bmod 11$ and, as expected, $d_{(9,2)}(3) = 5$ since $9 \times 5 \equiv 1 \bmod 11$ (see below) and $5 \times (3-2) \equiv 5 \bmod 11$.

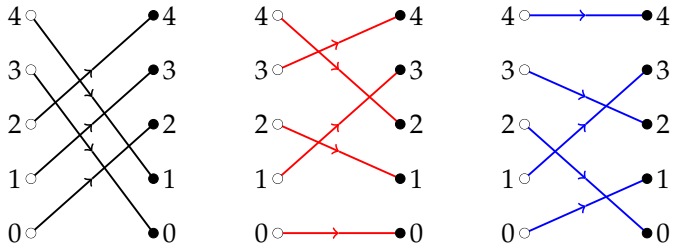
To find b, the multiplicative inverse of a in $\mathbb{Z}_q$, you can either do an exhaustive search, or run Euclid's algorithm to find b and s such that $ab + qs = 1$; then $ab \equiv 1 \bmod p$.

For instance, in $\mathbb{Z}_{11}$, to find 9^{-1}, we run Euclid's Algorithm getting $11 = 9 + 2$ and then $9 = 4 \times 2 + 1$, ending with the expected highest common factor of 1. Hence

$$1 = 9 - 4 \times 2 = 9 - 4 \times (11 - 9) = 5 \times 9 - 4 \times 11$$

and so $9^{-1} = 5$.

Exercise 4.3. The diagrams below show three encryption functions from the affine cipher when $q = 5$. What are the keys?



In Question 2(c) on Problem Sheet 3 you are asked to show that the affine cipher has perfect secrecy. So in a precise sense, observing a ciphertext tells Eve nothing about the plaintext. What can she learn about the key?

Exercise 4.4. Consider the affine cipher on $\mathbb{Z}_5$.

(i) Suppose that Mark knows that $e_{(a,c)}(1) = 3$. What does he learn about the key? What happens if he later learns $e_{(a,c)}(2)$?

(ii) Suppose that Eve observes the ciphertext 3. If she knows the plaintext is either 1 or 2 with equal probability, what does she learn about the key? (Compare Example 3.8(b) and the Group Work in Week 1.)

Attack models. In each of the *attack models* below, we suppose that Alice sends ciphertexts to Bob encrypted using the key $k \in \mathcal{K}$. The aim of the adversary (Eve or Mark) is to determine the plaintext and/or some information about k.

- *Known ciphertext*. Eve knows $e_k(x) \in \mathcal{C}$.
- *Known plaintext and ciphertext*. Mark knows $x \in \mathcal{P}$ and $e_k(x) \in \mathcal{C}$.
- *Chosen plaintext*. Mark may choose any $x \in \mathcal{P}$ and is given the encryption $e_k(x)$.
- *Chosen ciphertext*. Mark may choose any $y \in \mathcal{C}$ and is given the decryption $d_k(y)$.

Each attack model has a generalization where the adversary observes or chooses multiple plaintexts and/or ciphertexts.

Remark 4.5.

(1) In Example 2.5 we saw that (almost all) of the key in a substitution cipher can be deduced from a sufficiently long ciphertext. So the substitution cipher is broken by a *known ciphertext attack*.

(2) All the cryptosystems so far are broken by a *chosen plaintext attack*. By the general version of Example 4.4, the affine cipher requires two choices of plaintext, and by Question 4 on Sheet 2, the substitution cipher and the Vigenère cipher just one.

Exercise: How many chosen plaintexts does it take to break the numeric one-time pad?

(3) In Parts B and C we will see modern stream and block ciphers where it is believed to be computationally hard to find the key even allowing *unlimited* choices of plaintexts in a *chosen plaintext attack*.

One-time pad. Fix $n \in \mathbb{N}$. Let $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$ be the Roman alphabet. The *one-time pad* is a cryptosystem with $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathcal{A}^n$. You should think of $\mathcal{A}^n$ as all strings of length n. Thus $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}| = 26^n$. The encryption functions are defined by

$$e_k(x) = (x_0 + k_0, x_1 + k_1, \ldots, x_{n-1} + k_{n-1})$$

where, as in the Vigenère cipher (see Example 2.10), $x_i + k_i$ is computed by converting x_i and k_i to numbers and adding modulo 26. (Note, as before, we number positions from 0.) Thus the one-time pad is the Vigenère cipher when the key has the same length as the plaintext.

To make the cryptosystem practical, we assume that each key is used with the same probability.

Example 4.6. Suppose that $n = 8$. Of the 26^8 keys, suppose (by a $1/26^8$ chance) `zyxwvuts` is chosen. Then

$$e_{\mathtt{zyxwvuts}}(\mathtt{goodwork}) = \mathtt{fmlzrikc}$$

as shown in the table below.

i	0	1	2	3	4	5	6	7
x_i	g	o	o	d	w	o	r	k
	6	14	14	3	22	14	17	10
k_i	z	y	x	v	w	u	t	s
	25	24	23	22	21	20	19	18
$x_i + k_i$	5	12	11	25	17	8	10	2
	f	m	l	z	r	i	k	c

The one-time pad is closely related to the numeric one-time pad in Exercise 3.5. The following proposition is a corollary of Proposition 3.14. It could also be proved using the same strategy as Question 2 on Sheet 2 (as can this proposition).

Proposition 4.7. *The one-time pad has perfect secrecy.*

By the proposition, the one-time pad is secure against a known ciphertext attack with *one* ciphertext.

Example 4.8. The spy-master Alice and her agent Bob have agreed to use the one-time pad, with a key, chosen in advance uniformly at random. By Kerckhoffs's Principle, all this is known to Eve. Eve does not know that their key is $k = \texttt{atcldqezyomuua}$.

- Alice sends $e_k(\texttt{leaveinstantly}) = \texttt{lxcghyrrroznfy}$ to Bob.

Bob decrypts $\texttt{lxcghyrrroznfy} - \texttt{atcldqezyomuua} = \texttt{leaveinstantly}$. As expected from perfect secrecy, the ciphertext reveals nothing new about the plaintext. For example,

$$x = \texttt{gototheairport} \iff k = y - \texttt{gototheairport} = \texttt{fjjsornrjxkzof}$$
$$x = \texttt{meetmeonbridge} \iff k = y - \texttt{meetmeonbridge} = \texttt{ztynvudeqxrkzu}$$

and so on. Since each key is equally likely, as already seen in Proposition 4.7, Eve learns nothing about the plaintext.

If, for instance, the plaintext x' is very unlikely, then Eve now believes that the key $y - x'$ is very unlikely. So as in the Week 1 Group Work, Example 3.8 and Example 4.4, Eve learns something about the key. More precisely, $\mathbf{P}[K = k|Y = y] = \mathbf{P}[X = k - y|Y = y] = \mathbf{P}[X = k - y]$, where the final equality uses perfect secrecy. Provided the one-time pad is only used once, this is not a problem.

Bob now makes a fatal mistake, and re-uses the key k in his reply.

- Bob sends $e_k(\texttt{goingeasttrain}) = \texttt{ghkyjuerrhducn}$ to Alice.

Eve now has ciphertexts $k + \texttt{leaveinstantly} = \texttt{lxcghyrrroznfy}$ and $k + \texttt{goingeasttrain} = \texttt{ghkyjuerrhducn}$. She subtracts them, working modulo 26 in each position, to obtain $\Delta = \texttt{fqsiyenaahwtdl}$. Note that Δ does not depend on k.[2]

The string Δ has the unusual property that there is an English message x' (Bob's reply) such that $\Delta + x'$ is another English message (Alice's plaintext). This property is so rare that Eve and her computer can fairly easily deduce x' and $\Delta + x'$, and, from either of these, the key k. The code used in the demonstration is here: `repl.it/@mwildon/OneTimePad2`.

Thus the one-time pad is broken by a known ciphertext attack with *two* ciphertexts, when the likely plaintexts are English messages.

The slides include an analogy between this example and the Alice and Bob exam mark (Example 1.2) where Eve can make inferences from two observed ciphertexts $y = x + k \bmod 100$ and $y' = x' + k \bmod 100$, encrypted using the same key.

Exercise 4.9. Break the one-time pad using a chosen plaintext attack.

[2]We write Δ, the capital Greek letter like 'D', to emphasise that it is a *difference*: we will see the difference attack on block ciphers in Part C of the course.

5. KEY UNCERTAINTY AND ENTROPY

We continue the theme of §4.

Question. How can we make precise the amount of information an attacker learns about the key from an observed ciphertext?

Motivation for Entropy. Suppose Bob picks $x \in \{0, 1, \ldots, 15\}$. How many yes/no questions does Alice need to guess x? In the Group Work for Week 3 you saw one simple strategy: ask Bob to write x in binary as $x_3x_2x_1x_0$; then Alice asks about each bit in turn: 'Is $x_0 = 1$?', 'Is $x_1 = 1$?', 'Is $x_2 = 1$?', 'Is $x_3 = 1$?'.

Exercise 5.1. Explain why no questioning strategy can guarantee to use fewer than four questions.

Example 5.2. We consider the simpler game where Bob's number is in $\{0, 1, 2, 3\}$. Let p_x be the probability that Bob chooses x. (Alice knows Bob very well, so, as in Kerckhoffs's Principle, she knows these probabilities.) For each case below, how many questions does Alice need on average, if she chooses the best possible strategy?

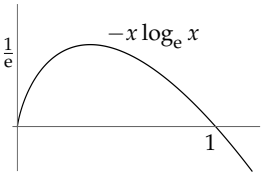
(a) $p_0 = p_1 = p_2 = p_3 = \frac{1}{4}$;
(b) $p_0 = \frac{1}{2}$, $p_1 = \frac{1}{4}$, $p_2 = \frac{1}{4}$, $p_3 = 0$;
(c) $p_0 = \frac{1}{2}$, $p_1 = \frac{1}{4}$, $p_2 = \frac{1}{8}$, $p_3 = \frac{1}{8}$;
(d) $p_0 = \frac{1}{8}$, $p_1 = \frac{1}{8}$, $p_2 = \frac{1}{4}$, $p_3 = \frac{1}{2}$.

Alice is most uncertain about Bob's number in (a), least uncertain in (b), and equally uncertain in (c) and (d). Remarkably, there is a way to make precise this 'degree of uncertainty', found by Shannon in 1948.[3]

Definition 5.3. Let $\mathcal{X}$ be a finite set. The *entropy* of a probability distribution p_x on $\mathcal{X}$ is

$$H(p) = -\sum_{x \in \mathcal{X}} p_x \log_2 p_x.$$

The *entropy* of a random variable X taking values in $\mathcal{X}$ is the entropy of the probability distribution $p_x = \mathbf{P}[X = x]$.



Note that $\log_2$ means logarithm to the base 2, so $\log_2 \frac{1}{2} = -1, \log_2 1 = 0$, $\log_2 2 = 1$, $\log_2 4 = 2$, and generally, $\log_2 2^n = n$ for each $n \in \mathbb{Z}$. If $p_x = 0$ then $-0 \log_2 0$ should be interpreted as $\lim_{p \to 0} -p \log_2 p = 0$; this limiting value can be seen from the graph in the margin.

[3] The story goes that Shannon asked von Neumann what he should call his measure of uncertainty, and von Neumann replied, '*You should call it entropy, for two reasons. In the first place your uncertainty function has been used in statistical mechanics under that name, so it already has a name. In the second place, and more important, no one really knows what entropy really is, so in a debate you will always have the advantage.*' While this may still be true, there is now a well-developed mathematical theory of entropy.

Exercise 5.4.

(i) Show that $H(p) = \sum_{x \in \mathcal{X}} p_x \log_2 \frac{1}{p_x}$, where if $p_x = 0$ then $0 \log_2 \frac{1}{0}$ is interpreted as 0.

(ii) Show that if p is the probability distribution in Exercise 5.2(b) then

$$H(p) = \tfrac{1}{2} \log_2 2 + \tfrac{1}{4} \log_2 4 + \tfrac{1}{4} \log_2 4 + 0 = \tfrac{3}{2}.$$

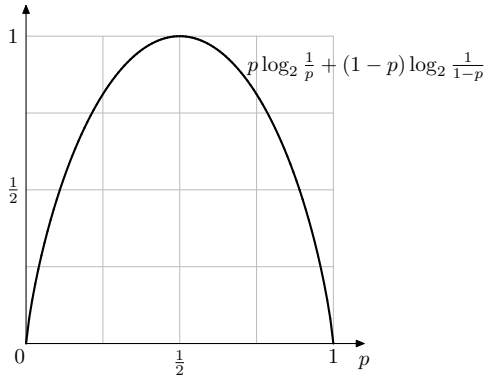
Show that in all three cases, $H(p)$ is the average number of questions, using the strategy found in this exercise.[4]

> **Informally.** A random variable has entropy h if and only if you can learn its value by asking about h well-chosen yes/no questions.

For this reason, entropy is often thought of as measured in bits. For example, the entropy of Bob's number in Example 5.2(a) is 2 bits.

Example 5.5.

(1) Suppose the random variable X takes two different values, with probabilities p and $1-p$. Then $H(X) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}$, as shown in the graph overleaf.



Thus the entropy of a single 'yes/no' random variable takes values between 0 and 1, with a maximum at 1 when the outcomes are equally probable.

(2) Suppose a cryptographic key K is equally likely to be any element of the keyspace $\mathcal{K}$. If $|\mathcal{K}| = n$ then $H(K) = \frac{1}{n} \log_2 n + \cdots + \frac{1}{n} \log_2 n = \log_2 n$. **This is often useful.**

(3) Consider the cryptosystem in Exercise 3.3(iii). Suppose that $\mathbf{P}[X = 0] = p$, and so $\mathbf{P}[X = 1] = 1 - p$, and that $\mathbf{P}[K = \text{red}] = r$, and so

[4] In general the entropy is only a lower bound for the average number of questions. For example, if $p_0 = \frac{1}{2}$, $p_1 = p_2 = p_3 = \frac{1}{6}$ then $H(p) = \frac{1}{2} \log_2 2 + 3\frac{1}{6} \log_2 6 = 1 + \frac{1}{2} \log_2 3 \approx 1.7925$. The best questioning strategy uses the Huffman code 1, 01, 000, 001 with average codeword length $\frac{1}{2}1 + \frac{1}{6}2 + \frac{1}{6}3 + \frac{1}{6}3 = \frac{11}{6} \approx 1.8333$. Huffman codes are part of MT341/441/5441 Classical Information Theory, in Term 2.

$\mathbf{P}[K = \text{black}] = 1 - r$. As in (1) we have

$$H(X) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}.$$

Exercise: show that $\mathbf{P}[Y = 1] = pr + (1-p)(1-r)$ and find $\mathbf{P}[Y = 0]$. Hence find $H(Y)$ when $r = 0, \frac{1}{4}, \frac{1}{2}$ and 1. Why is it not so surprising that $H(Y) > H(X)$?

Conditional entropy and key uncertainty.

Definition 5.6. Let K and Y be random variables taking values in finite sets $\mathcal{K}$ and $\mathcal{C}$, respectively.The *joint entropy* of K and Y is defined by

$$H(K,Y) = -\sum_{k \in \mathcal{K}} \sum_{y \in \mathcal{C}} \mathbf{P}[K = k \text{ and } Y = y] \log_2 \mathbf{P}[K = k \text{ and } Y = y].$$

The *conditional entropy of K given that* $Y = y$ is defined by

$$H(K|Y = y) = -\sum_{k \in \mathcal{K}} \mathbf{P}[K = k|Y = y] \log_2 \mathbf{P}[X = k|Y = y].$$

The *conditional entropy of K given* Y is defined by

$$H(K|Y) = \sum_{y \in \mathcal{C}} \mathbf{P}[Y = y] H(K|Y = y).$$

Note that $H(K,Y)$ is the entropy, as already defined, of the random variable (K,Y) taking values in $\mathcal{K} \times \mathcal{C}$ and $H(K|Y = y)$ is the entropy of the probability distribution of K *conditioned on* $Y = y$; i.e. the distribution $\mathbf{P}[K = k|Y = y]$ as k varies over $\mathcal{K}$.

You might also find it helpful to remember that $H(K|Y)$ is the expected value of $H(K|Y = y)$, as y varies over $\mathcal{C}$.[5]

Example 5.7. Consider the Caesar cryptosystem in which all 26 keys are equally likely. What is $H(K)$? Find $H(K|Y = \texttt{ACCB})$ and $H(K|Y = \texttt{NCYP})$, assuming Alice's message is a random English word.

Exercise 5.8. Let $r \in \mathbb{N}_0$. Consider the guessing game in which Bob's number X has probability distribution $(\frac{1}{2}, \frac{1}{2^{r+1}}, \ldots, \frac{1}{2^{r+1}})$.

(i) Using that the sum of probabilities is 1, what is the maximum possible number Bob might have chosen?

(ii) Show that $H(X) = 1 + r/2$.

(iii) Suppose that, as the first question, Alice asks

'Is your number 0?'

[5] If you have seen conditional expectation, please do not get confused into thinking that $H(K|Y)$ is a random variable. If you have not seen conditional expectation, please do not get confused by reading this footnote.

Let A be the answer. Show that $H(X|A = \text{'No'}) = r$. What is $H(X|A = \text{'Yes'})$? What is $H(X|A)$?

(iv) Show that $H(X|A) + H(A) = H(X, A)$. [*Hint:* $H(X, A) = H(X)$ since if you know X then you certainly know A.]

This example shows that we may have $H(X|Y = y) > H(X)$. It is true however that $H(X|Y) \leq H(X)$ (see the optional extras for this section). This inequality should be intuitive: knowing the further information in Y cannot, on average, make us more uncertain about X.

The most important property of conditional entropy is stated in the lemma below, and was seen in Exercise 5.8(iv). Intuitively 'the uncertainty of K and Y is the uncertainty of K given Y *plus* the uncertainty of Y'. (Now try reading this replacing 'uncertainty of' with 'information in'.)

Lemma 5.9 (Chaining Rule)**.** *Let K and Y be random variables. Then*

$$H(K|Y) + H(Y) = H(K, Y).$$

The proof of the Chaining Rule is examinable and will be given in a video, and 'live' if time permits. We need two further results to prove the main theorem of this section.

Lemma 5.10. *Let K and X be random variables. If K and X are independent then $H(K, X) = H(K) + H(X)$.*

For a proof (examinable) see Question 1 on Sheet 3 and the model answer.

Lemma 5.11. *Let Z be a random variable taking values in a set $\mathcal{Z}$. Let $f : \mathcal{Z} \to \mathcal{W}$ be a function. If f is injective then $H\big(f(Z)\big) = H(Z)$.* □

The immediate QED box means that I think this result is obvious. Intuitively, imagine rolling an unbiased die and let $Z \in \{1, 2, 3, 4, 5, 6\}$. By Exercise 5.5(2), $H(Z) = \log_2 6$. Let $f(z) = 2z$. If I tell you $f(Z)$, you can deduce Z (just take half!). So Z and $f(Z)$ have the same information and $H(f(Z)) = H(Z)$.[6]

Theorem 5.12 (Shannon, 1949)**.** *Take a cryptosystem in our usual notation. Then*

$$H(K|Y) = H(K) + H(X) - H(Y).$$

Proof (Examinable). By the Chaining Rule (Lemma 4.8), $H(K|Y) = H(K, Y) - H(Y)$. There is an injective function $f : \mathcal{K} \times \mathcal{P} \to \mathcal{K} \times \mathcal{C}$ defined by

$$f(k, x) = \big(k, e_k(x)\big).$$

[6] If you really want a formal proof, see the answer to Question 7(b) on Problem Sheet 3, which proves a more general result.

Clearly $f(K,X) = (K,Y)$. By Lemma 5.10, $H(K,Y) = H\big(f(K,X)\big) = H(K,X)$. Hence

$$\begin{aligned} H(K|Y) &= H(K,Y) - H(Y) \\ &= H(K,X) - H(Y) \\ &= H(K) + H(X) - H(Y) \end{aligned}$$

where the final line follows from Lemma 5.9, and our assumption that K and X are independent. □

We end with two applications of Shannon's Theorem.

English entropy and the one-time pad. As in Example 4.6, let $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$ be the alphabet and $\mathcal{P} = \mathcal{C} = \mathcal{A}^n$. To indicate that plaintexts and ciphertexts have length n, we write $X^{(n)}$ and $Y^{(n)}$ rather than X and Y.

We suppose only those strings that make good sense in English have non-zero probability. So if $n = 8$ then $\mathtt{abcdefgh}, \mathtt{goodwork} \in \mathcal{P}$ but $\mathbf{P}[X_8 = \mathtt{abcdefgh}] = 0$ whereas $\mathbf{P}[X_8 = \mathtt{goodwork}] > 0$.

Shannon estimated that the per-character redundancy of English plaintexts, with spaces, is about 3.2.

Let $R = 3.2$. If English plaintexts of length n had no redundancy, their per-character entropy would be $\log_2 26 \approx 4.7$. Therefore the per-character entropy of English is about $\log_2 26 - R \approx 1.5$, and

$$H(X^{(n)}) \approx (\log_2 26 - R)n \approx 1.5n.$$

Example 5.13 (Entropy for the one-time pad)**.** Suppose that all keys in $\mathcal{A}^n$ are equally likely. Then, as seen on Problem Sheet 3, all ciphertexts are equally likely, and by Example 5.5(2),

$$\begin{aligned} H(K) &= (\log_2 26)n \\ H(Y^{(n)}) &= (\log_2 26)n. \end{aligned}$$

We saw above that $H(X^{(n)}) \approx (\log_2 26 - R)n$. Therefore by Shannon's formula,

$$H(K|Y^{(n)}) = H(K) + H(X^{(n)}) - H(Y^{(n)}) = (\log_2 26 - R)n = H(X^{(n)}).$$

This is consistent with the remark after Example 4.6 on the probability distributions $\mathbf{P}[K = k|Y^{(n)} = y]$ (k varying) and p_x.

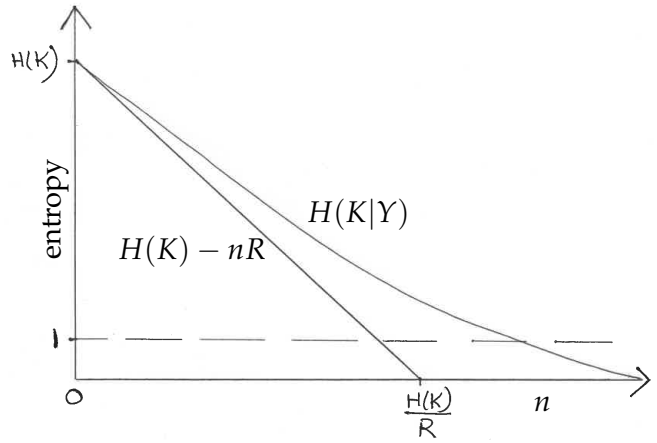
Unicity distance. In Example 5.13 we proved that for the one-time-pad $H(K|Y^{(n)}) = (\log_2 26 - R)n$ and that $H(K) = (\log_2 26)n$. Therefore

$$(\star\star) \qquad H(K|Y^{(n)}) = H(K) - Rn.$$

In the non-examinable extras for this part we give Shannon's argument that $(\star\star)$ should be a good approximation for $H(K|Y^{(n)})$ in any cryptosystem where $\mathcal{P} = \mathcal{C} = \mathcal{A}^n$, the messages are English texts, and keys are chosen uniformly at random. It works best when $\mathcal{K}$ is large and n is small.

Exercise 5.14. What is the largest length of ciphertext n for which $(\star\star)$ could hold with equality?

The graph below shows the expected behaviour of $H(K|Y)$.



Definition 5.15. The quantity $H(K)/R$ is the *unicity* distance of the cryptosystem.

If $H(K|Y^{(n)}) < 1$ then on average it takes less than one yes/no question to guess the key K. Therefore $(\star\star)$ predicts that most of the key is known when n is about the unicity distance of the cryptosystem.

Shannon's equation $(\star\star)$ predicts that the unicity distance for the substitution cipher is

$$\frac{\log_2 |\mathcal{K}|}{R} = \frac{\log_2(26!)}{R} \approx \frac{88.382}{3.200} = 27.619.$$

So 28 characters of ciphertext should, in theory, determine most of the key.

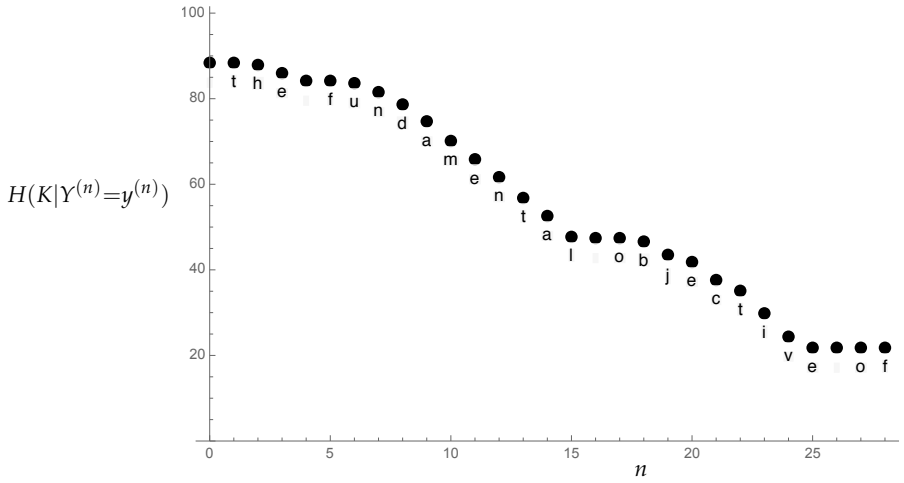
Example 5.16. The first 28 characters of the ciphertext in Example 2.5 are `KQX WJZRUHXZKUY GTOXSKPIX GW`. A computer search using a dictionary of about 70000 words (and all their subwords, for instance `tec` is a subword, but not a word) gives 6 possible decryptions of the first 24 letters. These include '`imo purgatorial hedonics`', '`the fundamental objectiv`' and '`tie fundamental povertys`'. Taking 25 letters,

'`the fundamental objective`'

is the only decryption consistent with the dictionary. This is in excellent agreement with Shannon's argument.

Since 10 characters do not appear in the first 28 letters of ciphertext, there are 10! possible keys; assuming equally probable keys we have $H(K|Y = y_{28}) = \log_2 10! = 21.791$. Nothing new about the key is learned after letter 25, so this is the value of the final 4 points in the graph of $H(K|Y^{(n)} = y^{(n)})$ for $0 \leq n \leq 28$ below.

Here $H(K|Y^{(n)} = y^{(n)} = \log_2 M^{(n)}$ where $M^{(n)}$ is the number of possible keys, computed on the assumption that all keys consisting with a decryption that is an English phrase with words of the correct length are equally likely. This is an oversimplification that will overestimate the true conditional entropy.



Exercise 5.17. The first dot, after no characters of ciphertext have been read, is at $\log_2 26! \approx 88.38195$. The next dot, after the first character has been read, showing $\log_2 \mathbf{P}[K|Y_1 = \text{'K'}]$ is also at this value. Why is this? [*Hint:* in the dictionary of 70000 words, there are words starting with every English letter; for instance, the '`xylophone`' makes its expected appearance.]

The remainder of the Part A notes have optional non-examinable extras. You are encouraged at least to read the first extra, which might be used for a quiz question or Group Work.

Extra: One idea in Shannon's argument for unicity distance.

Exercise 5.18. Suppose you ask each person in a large lecture room to state their number of siblings (for instance, an only child will reply 0), take the mean, and then add 1. Will the answer be a good estimator for the mean number of children in a family?

The answer is no! Because we always sample children, rather than families, we do not count any of the childless families. Worse, families with large numbers of children are disproportionately likely to have a child in the room. This 'selection bias' appears in Lemma 5.19 below.

Extra: Shannon's argument for unicity distance. The amazing insight in Shannon's proof of his Noisy Coding Theorem is that a good way to communicate on a binary channel that might corrupt the bits sent through it is to choose a large binary code *at random*. Shannon also worked out exactly how large this code can be so that the probability of error remains low. His proof will be seen in MT341/441/5441 Classical Information Theory.

Shannon introduced the analogous idea of the *random cryptosystem* in *Communication theory of secrecy systems*, Bell Systems Technical Journal **28** (1949) 656–715. Fix a set $\mathcal{P}$ of plaintexts, a set $\mathcal{C}$ of ciphertexts of the same size as $\mathcal{P}$, and a keyspace $\mathcal{K}$. For each $k \in \mathcal{K}$, choose a random bijection $e_k : \mathcal{P} \to \mathcal{C}$ as the encryption function. (Note that although the bijection is random, this random choice is made once and for all, right now. As in Kerckhoffs's Principle, everyone knows the encryption functions.)

As a simple model for English plaintexts, we fix a partition of $\mathcal{P}$

$$\mathcal{P} = \mathcal{P}_{\text{common}} \cup \mathcal{P}_{\text{rare}}$$

into two disjoint sets. We suppose that each common plaintext is sent with equal probability $1/|\mathcal{P}_{\text{common}}|$. Thus rare plaintexts are never sent, and each common plaintext is equally likely.

Suppose a plaintext is chosen at random and encrypted to the ciphertext $y \in \mathcal{C}$ by a key, chosen equiprobably from $\mathcal{K}$. Define

$$g(y) = \big|\{k \in \mathcal{K} : e_k(x) = y \text{ for some common plaintext } x\}\big|.$$

Equivalently, $g(y)$ is the number of keys k such that the decryption $e_k^{-1}(y)$ is common. Since y is the encryption of a common plaintext, we know that $g(y) \geq 1$. Since the keys are equiprobable, when Eve observes $y \in \mathcal{C}$, her uncertainty in the key is $\log_2 g(y)$. That is, $H(K|Y = y) = \log_2 g(y)$.

Let $|\mathcal{P}_{\text{common}}|/|\mathcal{P}| = c$ be the proportion of common plaintexts.

Lemma 5.19. $g(y) \sim 1 + \text{Bin}(|\mathcal{K}| - 1, c)$.

Proof. Suppose y was obtained by choosing $x^\star \in \mathcal{P}_{\text{common}}$ and $k^\star \in \mathcal{K}$, so $y = e_{k^\star}(x^\star)$. Since the encryption functions were chosen at random, for each $k \in \mathcal{K}$, with $k \neq k^\star$, the probability is c that $e_k^{-1}(y)$ is a common plaintext. Hence the number of such k is distributed as $\text{Bin}(|\mathcal{K}| - 1, c)$. Now add 1 to count $k^\star$. □

By the formula for conditional entropy,

$$\begin{aligned} H(K|Y) &= \sum_{y \in \mathcal{C}} H(K|Y = y)\mathbf{P}[Y = y] \\ &= \sum_{m \geq 1} \binom{|\mathcal{K}| - 1}{m - 1} c^{m-1}(1 - c)^{|\mathcal{K}|-m} \log_2 m \\ &= \frac{1}{c|\mathcal{K}|} \sum_{m \geq 0} \binom{|\mathcal{K}|}{m} c^m (1 - c)^{|\mathcal{K}|-m} m \log_2 m \end{aligned}$$

where we used $\binom{|\mathcal{K}|-1}{m-1} = \binom{|\mathcal{K}|}{m}\frac{m}{|\mathcal{K}|}$. The sum on the right-hand side is $\mathbb{E}[Z \log_2 Z]$, where $Z \sim \mathrm{Bin}(|\mathcal{K}|, c)$. When $|\mathcal{K}|$ is large compared to $|\mathcal{P}|$, Z is likely to be near its mean $c|\mathcal{K}|$, so $\mathbf{E}[Z \log_2 Z] \approx c|\mathcal{K}| \log_2(c|\mathcal{K}|)$. Hence

$$H(K|Y) \approx \frac{1}{c|\mathcal{K}|} c|\mathcal{K}| \log_2(c|\mathcal{K}|) = \log_2 |\mathcal{K}| + \log_2 c.$$

For English plaintexts of length n, we saw before Example 4.12 that $H(X^{(n)}) \approx (\log_2 26 - R)n$. Therefore a reasonable guess for $|\mathcal{P}_{\text{common}}|$ is $2^{(\log_2 26 - R)n}$. With this value, $\log_2 c = \log_2 |\mathcal{P}_{\text{common}}| - \log_2 26^n = -Rn$ and $H(K|Y) \approx \log_2 |\mathcal{K}| - Rn$, as in ($\star\star$).

Extra: An entropy inequality. We argued above on page 28 that if X is a random variable taking values in a set $\mathcal{X}$ and Y is a random variable taking values in a set $\mathcal{Y}$ then $H(X|Y) \leq H(X)$. Recall that, by definition, $H(X|Y) = \sum_{y \in \mathcal{Y}} \mathbf{P}[Y = y] H(X|Y = y)$. There would be an easy proof if $H(X|Y = y) \leq H(X)$, but this is not the case in general: see Example 5.8. Instead we must use an important inequality from information theory, which is used much more in the Classical Information Theory course in Term 2.

Lemma 5.20 (Gibbs' Inequality)**.** *Let $(p_1, \ldots, p_m)$ and $(q_1, \ldots, q_m)$ be probability distributions. Then $-\sum_{j=1}^m p_j \log_2 p_j \leq -\sum_{j=1}^m p_j \log_2 q_j$.*

We interpret the right-hand side as ∞ if $q_j = 0$ for some j such that $p_j \neq 0$. Incidentally Gibbs is mainly known for his scientific work: you might have heard of 'Gibbs free energy'.

We now use Gibbs' Inequality to prove that $H(X|Y) \leq H(X)$.

Proof. By the Chaining Rule, $H(X|Y) = H(X, Y) - H(Y)$. It is therefore equivalent to prove that $H(X, Y) \leq H(X) + H(Y)$. The left-hand side is the entropy of the probability distribution defined by

$$p_{(x,y)} = \mathbf{P}[X = x \text{ and } Y = y].$$

The right-hand side is the entropy of the probability distribution $q_{(x,y)} = \mathbf{P}[X = x]\mathbf{P}[Y = y]$; as motivation, note that q agrees with p if and only if X and Y are independent, in which case by Lemma 5.10, $H(X) + H(Y) = H(X, Y)$. By Gibbs' Inequality

$$-\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{(x,y)} \log_2 p(x, y) \leq -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{(x,y)} \log_2 q(x, y).$$

The right-hand side is

$$\begin{aligned} &-\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \mathbf{P}[X = x \text{ and } Y = y] \log_2 \mathbf{P}[X = x]\mathbf{P}[Y = y] \\ &\qquad = -\sum_{x \in \mathcal{X}} \Big(\sum_{y \in \mathcal{Y}} \mathbf{P}[X = x \text{ and } Y = y] \Big) \log_2 \mathbf{P}[X = x] \\ &\qquad\quad -\sum_{y \in \mathcal{Y}} \Big(\sum_{x \in \mathcal{X}} \mathbf{P}[X = x \text{ and } Y = y] \Big) \log_2 \mathbf{P}[Y = x] \end{aligned}$$

$$= -\sum_{x\in\mathcal{X}} \mathbf{P}[X=x]\log_2 \mathbf{P}[X=x] - \sum_{y\in\mathcal{Y}} \mathbf{P}[Y=y]\log_2 \mathbf{P}[Y=y]$$
$$= H(X) + H(Y).$$

Since the left-hand side is $H(X,Y)$, we have $H(X,Y) \leq H(X) + H(Y)$, as required. □

An extended version of Gibbs' Inequality states that equality holds in Lemma 5.20 if and only if $p_j = q_j$ for all j.

Exercise 5.21. Use this extended version to show that $H(X|Y) \leq H(X)$ with equality if and only if X and Y are independent.

A motivated proof of the extended version of Gibbs' Inequality. If $m = 1$ then $p_1 = q_1 = 1$ and both sides are 0, se may assume that $m \geq 2$. As in the shorter proof, we may also assume that $p_j > 0$ for all j, since if $p_j = 0$ then both summands $-p_j \log_2 p_j$ and $-p_j \log_2 q_j$ are zero. Let

$$G(q) = -\sum_{i=1}^{m} p_j \log_2 q_j.$$

We consider how $G(q)$ changes as we vary q over the set $\{(q_1,\ldots,q_m) \in \mathbb{R}^m : q_j \geq 0, \sum_{i=1}^m q_j = 1\}$ of probability distributions. It suffices to show that $G(q)$ is minimized uniquely when $q_j = p_j$ for all i. Since the set of probability distributions is closed and bounded, a minimum exists. Suppose it is $q^\star$. We have $0 < q_j^\star < 1$ for each i, since $-p_j \log_2 q_j \to \infty$ as $q_j \to 0$.

Exercise 5.22. Let $1 \leq i < j \leq s$. Show that on the line through $q^\star$ where we slightly increase q_i and slightly decrease q_j, staying in the set of probability measures, the values of $G(q)$ are given by

$$g(t) = G(q_1^\star,\ldots,q_i^\star + t,\ldots,q_j^\star - t,\ldots,q_s^\star)$$
$$= p_i \log_2(q_i^\star + t) + p_j \log_2(q_j^\star - t) + \sum_{i\neq j,k} p_i \log_2 q_i^\star$$

Show that

$$g'(t) = \frac{p_i}{q_i^\star \log_e 2} - \frac{p_j}{q_j^\star \log_e 2}.$$

Since $q^\star$ is a minimum of G, we have $g'(t) = 0$. Deduce from this that $p_i/q_i^\star = p_j/q_j^\star$ for all i and j and hence that $q^\star = p$.

Hence $-\sum_{j=1}^m p_j \log_2 p_j \leq G(q)$ with equality if and only if $q = p$. This is Gibbs' Inequality. □

If you have seen Lagrange multipliers you might recognise that this proof uses the same method.

(B) Stream ciphers

6. Linear Feedback Shift Registers

Computers are deterministic: given the same inputs, you always get the same answer.

> **Question.** How can we get a sequence that 'looks random' out of a deterministic algorithm? How can we use it to encrypt a plaintext?

Reminder of binary. Recall that $\mathbb{F}_2$ is the finite field of size 2 with elements the *bits* (short for *bi*nary digi*ts*) 0, 1. Addition and multiplication are defined modulo 2, so

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

By definition, $\mathbb{F}_2^n$ is the set of n-tuples $(x_0, x_1, \ldots, x_{n-1})$ where each x_i is a bit 0 or 1. For brevity we may write this tuple as $x_0x_1 \ldots x_{n-1}$. As usual in this course, we number positions from 0 up to $n-1$. It is usual to refer to elements of $\mathbb{F}_2^n$ as *binary words* of length n.

Exercise 6.1. Write down 15 bits in a circle so that, reading the cycle clockwise, every non-zero binary word of length 4 appears exactly once. How many 0s do you use? How many 1s do you use?

Definition of LFSRs.

Definition 6.2.

(i) Let $\ell \in \mathbb{N}$. A *linear feedback shift register* of *width* ℓ with *taps* $T \subseteq \{1, 2, \ldots, \ell\}$ is a function $F : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell$ of the form

$$F\big((x_0, x_1, \ldots, x_{\ell-2}, x_{\ell-1})\big) = (x_1, \ldots, x_{\ell-1}, \sum_{t\in T} x_{\ell-t}).$$

(ii) The function $f : \mathbb{F}_2^\ell \to \mathbb{F}_2$ defined by $f(x) = \sum_{t\in T} x_{\ell-t}$ is called the *feedback function*.

(iii) The *keystream* for $k \in \mathbb{F}_2^\ell$ is the sequence $k_0, k_1, \ldots, k_{\ell-1}, k_\ell, k_{\ell+1}, \ldots$, where for each $s \geq \ell$ we define

$$k_s = \sum_{t\in T} k_{s-t}$$

Equivalently, $k_s = f\big((k_{s-\ell}, k_{s-\ell+1}, \ldots, k_{s-1})\big)$ and so

$$F\big((k_{s-\ell}, k_{s-\ell+1}, \ldots, k_{s-1})\big) = (k_{s-\ell+1}, \ldots, k_{s-1}, k_s).$$

Thus the LFSR function F shifts the bits in the first $\ell - 1$ positions left (forgetting the very first), and puts a new bit, defined by its feedback

function, into the rightmost position. Taking all these rightmost positions gives the keystream. We call this the **Very Useful Property**:

(**VUP**) $$F^s\big((k_0, k_1, \ldots, k_{\ell-1})\big) = (k_s, k_{s+1}, \ldots, k_{s+\ell-1}).$$

Here F^s is the function defined by applying F a total of s times. To simplify notation, one may write $F^s(k_0, k_1, \ldots, k_{\ell-1})$ for the left-hand side, omitting one set of parentheses.

Example 6.3. The LFSR F of width 4 with taps $\{3, 4\}$ is defined by

$$F\big((x_0, x_1, x_2, x_3)\big) = (x_1, x_2, x_3, x_0 + x_1).$$

(i) Solving the equation $F\big((x_0, x_1, x_2, x_3)\big) = (y_0, y_1, y_2, y_3)$ shows that F has inverse

$$F^{-1}\big((y_0, y_1, y_2, y_3)\big) = (y_0 + y_3, y_0, y_1, y_2).$$

(ii) The keystream for the key $k = \mathbf{0111}$ is

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

repeating from position 15 onwards: $k_s = k_{s+15}$ for all $s \in \mathbb{N}_0$. (As a notational guide we sometimes use bold letters for the initial key and its repeats: it is entirely optional.)

(iii) *Exercise:* observe that $k' = 0001$ appears as $k_5k_6k_7k_8$ in the keystream. Find the keystream when the LFSR is *started* with k'.

(iv) By the (**VUP**), starting with $k = 0111$, we have $k_1k_2k_3k_4 = F(k) = 1111$ and $k_2k_3k_4k_5 = F^2(k) = 1110$. The full sequence k, $F(k)$, $F^2(k)$, $F^3(k)$,..., $F^{14}(k)$, $F^{15}(k)$ is

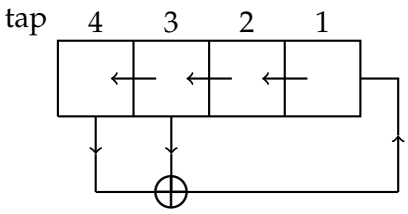
$$\mathbf{0111} \mapsto 1111 \mapsto 1110 \mapsto 1100 \mapsto 1000 \mapsto 0001 \mapsto 0010 \mapsto 0100$$
$$\mapsto 1001 \mapsto 0011 \mapsto 0110 \mapsto 1101 \mapsto 1010 \mapsto 0101 \mapsto 1011 \mapsto \mathbf{0111}$$

with $F^{15}(k) = k$. Observe that, as expected from (**VUP**), the rightmost bits in each $F^s(k)$, namely $1, 1, 0, 0, 0, 0, 1 \ldots$ are the keystream for **0111**, starting from $k_3 = 1$.

(v) We say that $x'_0x'_1 \ldots x'_{n-1}$ is a *cyclic shift* of $x_0x_1 \ldots x_{n-1}$ if there exists r such that $x_rx_{r+1} \ldots x_{n-1}x_0 \ldots x_{r-1} = x'_0x'_1 \ldots x'_{n-1}$.

Exercise: Is every keystream generated by F a cyclic shift of the keystream for 0001?

In the cryptographic literature it is conventional to represent LFSRs by circuit diagrams, such as the one below showing F of width 4 with taps $\{3, 4\}$. By convention $\oplus$ denotes addition modulo 2, implemented in electronics by the XOR gate.



The word 'register' in LFSR refers to the boxed memory units storing each bit.

Cryptosystem defined by an LFSR.

Definition 6.4. Let F be an LFSR of width ℓ and let $n \in \mathbb{N}$. The *cryptosystem defined by* F has $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^n$ and keyspace $\mathcal{K} = \mathbb{F}_2^\ell$. The encryption functions are defined by

$$e_k(x) = (k_0, k_1, \ldots, k_{n-1}) + (x_0, x_1, \ldots, x_{n-1})$$

for each $k \in \mathcal{K}$ and $x \in \mathcal{P}$.

Thus, like the one-time pad, the ciphertext is obtained by addition to the plaintext. But unlike the one-time pad, the key is usually much shorter than the plaintext.

Exercise 6.5. Define the decryption function $d_k : \mathbb{F}_2^n \to \mathbb{F}_2^n$.

Invertible LFSRs and periods.

Exercise 6.6. Let H be the LFSR of width 3 with taps $\{1, 2\}$. Show that H is not invertible and check that 111**011**011**011**011 ... is a keystream of H, ending in the cycle **011**011 ....

This exercise and Example 6.3(i) suggest the general result: an LFSR of width ℓ is invertible if and only if ℓ is one of the taps. The steps in a proof are indicated in Question 3 of Sheet 4.

Exercise 6.7. Let G be the LFSR of width 4 with taps $\{1, 2, 4\}$.

(a) Find the keystreams for the keys 0001 and 0010.
(b) Which words of length 4 do not appear in either keystream?
(c) Find all keystreams generated by this LFSR.

For cryptographic purposes, an invertible LFSR is used, and we want the keystream to be as long as possible before it repeats. For invertible LFSRs, we now show this repeat must be 'from the start'.

Fix a non-zero key $k \in \mathbb{F}_2^\ell$ and consider the binary words $F^s(k)$ for $s \in \mathbb{N}_0$. **Mini-exercise:** why are they all non-zero? As in Example 6.3(iv), we make a chain

$$k \mapsto F(k) \mapsto F^2(k) \mapsto \ldots \mapsto F^s(k) \mapsto \ldots \mapsto F^{s'}(k) \mapsto \ldots.$$

Since there are $2^\ell - 1$ non-zero binary words of length ℓ, and

$$k, F(k), \ldots, F^{2^\ell - 1}(k)$$

has 2^ℓ words, there exist r, r' with $0 \leq r < r' < 2^\ell$ such that $F^r(k) = F^{r'}(k)$. Now applying F^{-r} we get $k = F^{r'-r}(k)$. Hence, by (**VUP**),

$$k_0 k_1 \ldots k_{\ell-1} = k_{r'-r} k_{r'-r+1} \ldots k_{r'-r+\ell-1}$$

and the keystream repeats after at most $r' - r < 2^\ell$ positions.

Definition 6.8. Let F be an invertible LFSR.

(i) We define the *period* of a keystream $k_0, k_1, \ldots$ generated by F to be the least $p \in \mathbb{N}$ such that $k_{s+p} = k_s$ for all $s \in \mathbb{N}_0$.

(ii) We define the *period* of F to be the least $P \in \mathbb{N}$ such that $F^P = \mathrm{id}$, the identity function.

By the argument before the definition, each p is well-defined and if F has width ℓ then $p \leq 2^\ell - 1$. By (**VUP**), P is the lowest common multiple of the periods of the keystreams of F. (See the optional Question 5 on Problem Sheet 4 for details.) In fact P is the maximum period of a keystream. (See the optional Question 7 on Problem Sheet 4: this follows easily from Lemma 5.4 in the M.Sc. course, and is part of Corollary 5.6.)

For example, the invertible LFSRs F and G in Example 6.3 and Exercise 6.7 have non-zero keystreams of periods 15 (the maximum possible) and $7, 7, 1, 1$ respectively. The LFSR periods are 15 and 7, respectively.

7. Keysteams and randomness

Question. How random is the keystream generated by an invertible LFSR? What do we mean by 'random' anyway?

We saw before Definition 6.8 that the maximum possible period of a keystream of an LFSR of width ℓ is $2^\ell - 1$. Given any non-zero $k \in \mathbb{F}_2^\ell$, the first $2^\ell - 1$ positions of the keystream for k are the *generating cycle* for k. (The term '*m-sequence*' is also used.) Thus

(†) $$k_{2^\ell - 1 + s} = k_s \text{ for all } s \in \mathbb{N}.$$

Exercise 7.1. Let F be the LFSR of width 4 with taps $\{3, 4\}$ and period $15 = 2^4 - 1$ seen in Example 6.3. It has the maximum possible period for its width. The keystream for $k = (1, 1, 0, 0)$ can be obtained by reading the keystream in Example 6.3 from 1100. It is

$$(1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0 \ldots).$$

Correspondingly, by the Very Useful Property (**VUP**),

$$F(1,1,0,0) = (1,0,0,0),\ F^2(1,1,0,0) = (0,0,0,1), \ldots,\ F^{14}(1,1,0,0) = (1,1,1,0)$$

and $F^{15}(1,1,0,0) = (1,1,0,0)$. By taking the first 15 positions we get the generating cycle

$$\underset{k_0\ k_1\ k_2\ k_3\ k_4\ k_5\ k_6\ k_7\ k_8\ k_9 k_{10} k_{11} k_{12} k_{13} k_{14}}{(1,1,0,0,0,1,0,0,1,1,0,1,0,1,1)}$$

Since the keystream period is 15, $k_{15+s} = k_s$ for all $s \in \mathbb{N}_0$, as seen in (†).

(a) Find all the positions s such that

$$(k_s, k_{s+1}, k_{s+2}, k_{s+3}) = (0,1,1,1).$$

(b) What is the only element of $\mathbb{F}_2^4$ *not* appearing in the keystream for $(0,0,0,1)$?

(c) Why is the generating cycle for $(0,1,1,1)$ a cyclic shift of the generating cycle for $(1,1,0,0)$?

(d) Find all the positions s such that $(k_s, k_{s+1}, k_{s+2}) = (0,1,1)$. How many are there? [*Hint:* you do something similar in the Group Work for Week 5.]

(e) Repeat (d) changing $(0,1,1)$ to $(0,0,1)$, $(0,0,0)$ and then to $(0,1)$, $(1,1)$, $(1,0)$ and $(0,0)$. Explain the pattern.

Proposition 7.2. *Let F be an invertible LFSR of width ℓ with a keystream of period $2^\ell - 1$. Let $k \in \mathbb{F}_2^\ell$ be non-zero and let $(k_0, k_1, \ldots, k_{2^\ell - 2})$ be its generating cycle. We consider starting positions s within this cycle, so $0 \leq s < 2^\ell - 1$.*

(a) *For each non-zero $x \in \mathbb{F}_2^\ell$ there exists a unique s such that*

$$(k_s, \ldots, k_{s+\ell-1}) = x.$$

(b) *Given any non-zero $y \in \mathbb{F}_2^m$ where $m \leq \ell$, there are precisely $2^{\ell-m}$ positions s such that $(k_s, \ldots, t_{s+m-1}) = y$.*

(c) *There are precisely $2^{\ell-m} - 1$ positions s such that $(k_s, \ldots, k_{s+m-1}) = (0,0,\ldots,0) \in \mathbb{F}_2^m$.*

In particular, (b) and (c) imply that, in a generating cycle of an invertible LFSR of width ℓ and maximal possible period, there are $2^{\ell-1}$ ones and $2^{\ell-1} - 1$ zeros. How many times do 00, 01, 10 and 11 appear?

Exercise 7.3. Write down a sequence of 33 bits, fairly quickly, but trying to make it seem random. Count the number of zeros and the number of ones. Now count the number of adjacent pairs 00, 01, 10, 11. Does your sequence still seem random?

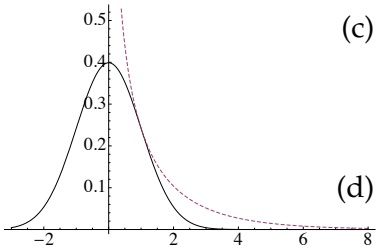
Random sequences of length 33 will have, on average, $16\frac{1}{2}$ zeros and ones, and 8 of each pair 00, 01, 10, 11. But because they are random, some will have more, and some less. At what point should we suspect that the sequence is not truly random?

Here we answer this question for the first test in Exercise 7.3, counting the number of zeros and ones. This is the *monobit test*.

Exercise 7.4 (Monobit Test)**.** Let M_0 be the number of zeros and let M_1 be the number of ones in a binary sequence $B_0, B_1, \ldots, B_{n-1}$ of length n.

(a) Explain why if the bits are random we would expect that M_0 and M_1 both have the $\mathrm{Bin}(n, \frac{1}{2})$ distribution.

(b) Show that the χ^2 statistic with (a) as null hypothesis is $(M_0 - M_1)^2/n$.



(c) A sequence with $n = 80$ has 50 zeros. Does this suggest it is not truly random? [*Hint:* if $Z \sim N(0,1)$ then $\mathbf{P}[Z^2 \geq 3.841] \approx 0.05$ and $\mathbf{P}[Z^2 \geq 6.635] \approx 0.01$. The probability density functions for Z (solid) and Z^2 (dashed) are shown in the margin.]

(d) (For statisticians.) Is the test used in (c) one-tailed or two-tailed?

See Question 4 on Problem Sheet 5 for the analogous test looking at pairs of adjacent bits and the slides and videos for a quiz on the hypothesis testing framework.

Another interesting measure of randomness is the degree to which a sequence is correlated with a shift of itself.

Definition 7.5. Given $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1}) \in \mathbb{F}_2^n$ define

$$c_{\text{same}} = \big|\{i : x_i = y_i\}\big|$$
$$c_{\text{diff}} = \big|\{i : x_i \neq y_i\}\big|.$$

The *correlation* between x and y is $(c_{\text{same}} - c_{\text{diff}})/n$.

Exercise 7.6. Find the correlation between a generating cycle for the LFSR of width 3 with taps $\{2, 3\}$ and each cyclic shift of itself. Would your answer change if you used a different key to calculate the generating cycle?

More generally we shall prove the following proposition, which again shows that a generating cycle of an LFSR of maximum possible period for its width has a strong randomness property.

Proposition 7.7. *Let $(k_0, k_1, \ldots, k_{2^\ell-2})$ be a generating cycle of an LFSR of width ℓ and maximum possible period $2^\ell - 1$. Let $1 \leq r < 2^\ell - 1$. The correlation between $(k_0, k_1, \ldots, k_{2^\ell-2})$ and its proper cyclic shift*

$$(k_r, k_{r+1}, \ldots, k_{2^\ell-2}, k_0, \ldots, k_{r-1})$$

is $-\frac{1}{2^\ell-1}$.

Remark on proof. If you remember the idea of defining $u_s = k_s + k_{s+r}$ and that 'sames' and 'differents' correspond to 0s and 1s in the keystream for $(u_0, \ldots, u_{\ell-1})$, you should be able to reconstruct the entire proof.

Note that $(u_0, \ldots, u_{\ell-1})$ is a non-zero key because otherwise $k_0 = k_r, \ldots, k_{\ell-1} = k_{\ell+r-1}$, and so the keystream repeats after r positions, contradicting that its period is $2^\ell - 1$.

8. Non-linear stream ciphers

Question. Why is the LFSR cryptosystem weak? What can we do to improve it?

A general stream cipher takes a key $k \in \mathbb{F}_2^\ell$, for some fixed ℓ, and outputs a *keystream* $u_0u_1u_2\ldots$ of bits. For each $n \in \mathbb{N}$ there is a corresponding cryptosystem where, as in Definition 6.4, the encryption functions $e_k : \mathbb{F}_2^n \to \mathbb{F}_2^n$ are defined by

$$e_k(x) = (u_0, u_1, \ldots, u_{n-1}) + (x_0, x_1, \ldots, x_{n-1}).$$

Exercise 8.1. In the LFSR cryptosystem of Definition 6.4, the keystream $u_0u_1u_2\ldots$ is simply $k_0k_1k_2,\ldots$. Show how to find the key $(k_0, \ldots, k_{\ell-1})$ using a chosen plaintext attack.

One reason why this cryptosystem is weak is because every bit of internal state appears, unmodified, in the keystream.

Example 8.2. A way to avoid this weakness is to use two or more LFSR keystreams as the internal state of the stream cipher, adding them to create the output keystream. Some care is needed.

- Let F be the LFSR of width 4 with taps $\{3, 4\}$ of period 15.

The first 20 bits in the keystreams for F with keys $k = (0, 0, 0, 1)$ and $k' = (1, 1, 1, 1)$ sum to the sequence $(u_0, u_1, \ldots, u_{19})$ below:

$$\begin{array}{ll} k_i & 0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,0,0,0,1,0 \\ k_i^\star & 1,1,1,1,0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,0 \\ u_i & 1,1,1,0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,0,0 \\ & 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9 \end{array}$$

The generating cycle $0,0,0,1,\ldots,1,1,1,1$ is highlighted on its first appearance. We see that $(u_0, u_1, u_2, \ldots)$ is also generated by F: since it starts 1110, it is the keystream for $(1, 1, 1, 0)$.

(a) *Exercise:* Explain why this should have been expected. [*Hint:* the same linearity was used to prove Proposition 7.7.]

(b) *Exercise:* can the keys k and $k^\star$ be recovered from (u_0, u_1, u_2, u_3)? If so, explain how; if not, explain how an attacker given (u_0, u_1, u_2, u_3) can still decrypt ciphertexts encrypted by adding the keystreams for k and $k^\star$ to the plaintext.

- Let F' be the LFSR of width 3 with taps $\{2, 3\}$ of period 7.

The first 20 bits in the keystreams for F and F' with keys $k = (0, 0, 0, 1)$ and $k' = (0, 0, 1)$ and their sum $(u_0, u_1, \ldots, u_{19})$ are:

$$\begin{array}{ll} k_i & 0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,0,0,0,1,0 \\ k_i' & 0,0,1,0,1,1,1,0,0,1,0,1,1,1,0,0,1,0,1,1 \\ u_i & 0,0,1,1,1,1,0,1,0,0,0,0,0,0,1,0,1,0,0,1 \\ & 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9 \end{array}$$

Exercise: what is the period of $(u_0, u_1, u_2, \ldots)$?

The exercise is encouraging: combining the LFSRs creates a keystream with a much longer period than either individually.

The bad news is that the keystream $(u_0, u_1, u_2, \ldots)$ is generated by the LFSR of width 7 with taps $\{2, 4, 5, 7\}$.[7] So as in (b) above, the keystream $u_0u_1u_2\ldots$ is the keystream of a single LFSR. In particular an attacker who learns $(u_0, u_1, \ldots, u_6)$ can calculate the entire keystream and decrypt any further ciphertexts sent using k and k'.

To avoid this problem, modern stream ciphers use non-linear functions, such as multiplication. They also avoid using every bit of the internal state in the keystream.

Example 8.3. A *Geffe generator* is constructed using three LFSRs F, F' and G of widths ℓ, ℓ' and m, all with maximum possible period. Following Kerckhoff's Principle, the widths and taps of these LFSRs are public knowledge.

- Let $k_0k_1k_2\ldots$ and $k'_0k'_1k'_2\ldots$ be keystreams for F and F'
- Let $g_0g_1g_2\ldots$ be a keystream for G.

The *Geffe keystream* $(u_0, u_1, u_2, \ldots)$ is defined by

$$u_i = \begin{cases} k_i & \text{if } g_i = 0 \\ k'_i & \text{if } g_i = 1. \end{cases}$$

For example, if F and F' and their keystreams are as in Example 8.2 (so F has width 4, taps $\{3, 4\}$, F' has width 3, taps $\{2, 3\}$), and G is the LFSR of width 4 with taps $\{1, 4\}$ and $(g_0, g_1, g_2, g_3) = (0, 0, 0, 1)$ then, using colours to indicate which bit is used:

k_i	0,	0,	0,	1,	0,	0,	1,	1,	0,	1,	0,	1,	1,	1,	1,	0,	0,	0,	1,	0
k'_i	0,	0,	1,	0,	1,	1,	1,	0,	0,	1,	0,	1,	1,	1,	0,	0,	1,	0,	1,	1
g_i	0,	0,	0,	1,	1,	1,	1,	0,	1,	0,	1,	1,	0,	0,	1,	0,	0,	0,	1,	1
u_i	0,	0,	0,	0,	1,	1,	1,	1,	0,	1,	0,	1,	1,	1,	0,	0,	0,	0,	1,	1
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9

Exercise: give an upper bound on the period of $(u_0, u_1, u_2, \ldots)$, for this example, and in general.

The Geffe generator is much better than taking the sum of $k_0k_1k_2\ldots$ and $k'_0k'_1k'_2\ldots$. But it is vulnerable to a correlation attack.

Exercise: Assume that the keystreams $k_0k_1k_2\ldots$ and $k'_0k'_1k'_2\ldots$ have independent bits. Find $\mathbf{P}[k_s = u_s]$ for each $s \in \mathbb{N}_0$.[8]

[7] **M.Sc.** students will see the theoretical reason for this using annihilators in §5 of their course; §6 of the M.Sc. course is on the Berlekamp–Massey algorithm that gives another way to find these taps.

[8] Formally, this means $\mathbf{P}[K_s = U_s]$, where K_s and U_s are the random variables for the first keystream and the Geffe keystream, where, by our assumption K_s and U_s are independently and uniformly distributed on $\{0, 1\}$.

Up to a tiny error (it is even smaller than the $-1/2^{\ell-1}$ in Proposition 7.7), the independence assumption holds in practice. Thus the correlation between $k_0k_1k_2\ldots$ and $u_0u_1u_2\ldots$ is very nearly $\frac{3}{4}-\frac{1}{4}=\frac{1}{2}$. Recall that 0 corresponds to no correlation, 1 to equality in every position and -1 to inequality in every position.

Attack 8.4. *Suppose that n bits of the Geffe keystream are known. The attacker computes, for each candidate key $(v_0,v_1,\ldots,v_{\ell-1})\in\mathbb{F}_2^\ell$, the correlation between $v_0v_1\ldots v_{n-1}$ and $u_0u_1\ldots u_{n-1}$. If the correlation is not nearly $\frac{1}{2}$ then the candidate key is rejected. Otherwise it is likely that $(k_0,k_1,\ldots,k_{\ell-1})=(v_0,\ldots,v_{\ell-1})$.*

Exercise: in the example above, $\ell>\ell'$. Is it better to guess the key for F or for F'?

One can repeat Attack 8.4 to learn $(k'_0,k'_1,\ldots,k'_{\ell'-1})$. Overall this requires at most $2^\ell+2^{\ell'}$ guesses. This is a huge improvement on the $2^{\ell+\ell'+m}$ guesses required by trying every possible triple of keys. Question 1(b) on Sheet 6 suggests some ways to speed up finding k' once k is known.

An attack such as Attack 8.4 is said to be *sub-exhaustive* because it finds the key using fewer guesses than brute-force exhaustive search through the keyspace.

Extras for Part B. Adding up multiple bits reduces the bias seen in the Geffe generator. This is 'extra' for the MT362/462 course but examinable for MT5462. It is a nice application of the 'Piling-up' Lemma (Lemma 4.11) in the **M.Sc.** notes.

Example 8.5. Let F be the LFSR of width 5 with taps $\{3,5\}$ and let F' be the LFSR of width 6 with taps $\{2,3,5,6\}$. These have the maximum possible periods for their widths, namely $2^5-1=31$ and $2^6-1=63$. Fix $m\in\mathbb{N}$ and for each $i\geq m$, define

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

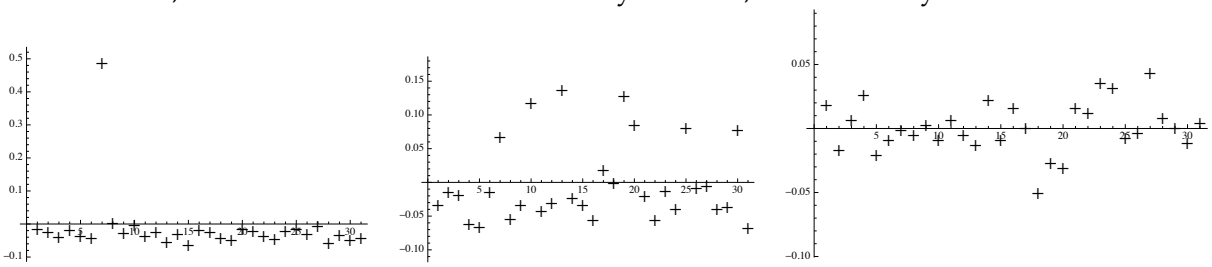
Note that there are m products in the sum. Define $u_s=0$ if $0\leq s<m-1$. The *m-quadratic stream cipher* is the cryptosystem defined using the keystream $u_0u_1\ldots u_{1023}$.

Taking $m=1$ gives a cipher like the Geffe generator: since $u_s=k_sk'_s$ we have $\mathbf{P}[u_s=k_s]=\frac{3}{4}$, giving a correlation of $\frac{1}{2}$. Attack 8.4 is effective.

For general m, the expected correlation between keystream of the m-quadratic stream cipher $u_0u_1u_2\ldots u_{1023}$ and the keystream $k_0k_1k_2\ldots k_{1023}$ of the LFSR of width 5 is about $\frac{1}{2^m}$. (This follows from the 'Piling-Up' lemma, Lemma 5.11 in the **M.Sc.** course.) Taking $m=5$, this makes the correlation attack ineffective because the difference between 0 correlation

and the correlation of $\pm\frac{1}{2^5}$ from a correct key guess cannot be detected with 2^{10} samples.

The graphs below show correlations for all 31 non-zero keys k when $m = 1$, $m = 3$ and $m = 5$. The correct key is 00111, or 7 in binary.



Notice that when $m = 3$ there are 7 'fake keys' with positive correlation, as well as the correct key. **M.Sc.** students are asked to explain this, in the case $m = 2$, in Question 3 on Problem Sheet 6.

Exercise 8.6. Unfortunately the m-quadratic cipher can still be attacked because the sum of two adjacent bits u_i and u_{i-1} in the keystream cancels out many of the quadratic terms. Use this to find a subexhaustive attack.

We end by looking at a modern stream cipher that, like the quadratic cipher, mixes multiplication and addition on multiple LFSRs. This combination gives a practical cipher with no known sub-exhaustive attacks.

Example 8.7 (TRIVIUM)**.** The building blocks are three LFSRs of widths 93, 84 and 111, with taps $\{66, 93\}$, $\{69, 84\}$ and $\{66, 111\}$. Let $x \in \mathbb{F}_2^{93}$, $y \in \mathbb{F}_2^{84}$, $z \in \mathbb{F}_2^{111}$ be the internal states. The registers are updated using the functions f, g and h, respectively, where

$$f(x, y, z) = z_0 + z_{111-66} + z_1 z_2 + x_{24}$$
$$g(x, y, z) = x_0 + x_{93-66} + x_1 x_2 + y_6$$
$$h(x, y, z) = y_0 + y_{84-69} + y_1 y_2 + z_{24}$$

For instance the x-register is updated using f, so in each step

$$(x_0, \ldots, x_{92}) \mapsto \big(x_1, \ldots, x_{92}, f(x, y, z)\big).$$

The keystream bit from each step is

$$x_0 + x_{93-66} + y_0 + y_{84-69} + z_0 + z_{111-66}.$$

Rather than use a 288-bit key, TRIVIUM uses a (secret) 80-bit key put in the x-register, and a (non-secret) 80-bit initialization vector put in the y-register. The remaining positions in the internal state start as 0, except for z_0, z_1, z_2 which start as 1.[9] (Exercise: why do this?) The first 1152 bits of the keystream are unusually biased, and so are discarded. This can be seen, for the earlier bits, using the implementation of TRIVIUM in the MATHEMATICA notebook on Moodle.

[9] See `http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf` for details: for consistency with the conventions in this part, the right-shifting registers in the formal specification have been converted to (equivalent) left-shifting registers, as in Definition 6.2, and in the specification on Wikipedia.

(C) Block ciphers

9. Feistel networks and DES

Question. What are the strongest cryptosystems in common use?

In a block cipher of *block size* n and *key length* ℓ, $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^n$, and $\mathcal{K} = \mathbb{F}_2^\ell$. Since $\mathcal{P} = \mathcal{C}$, by Exercise 3.3(iii), each encryption function e_k for $k \in \mathcal{K}$ is bijective, and the cryptoscheme is determined by the encryption functions.

In a typical modern block cipher, $n = 128$ and $\ell = 128$. Since most messages have more than n bits, they have to be split into multiple *blocks*, each of n bits, before encryption.

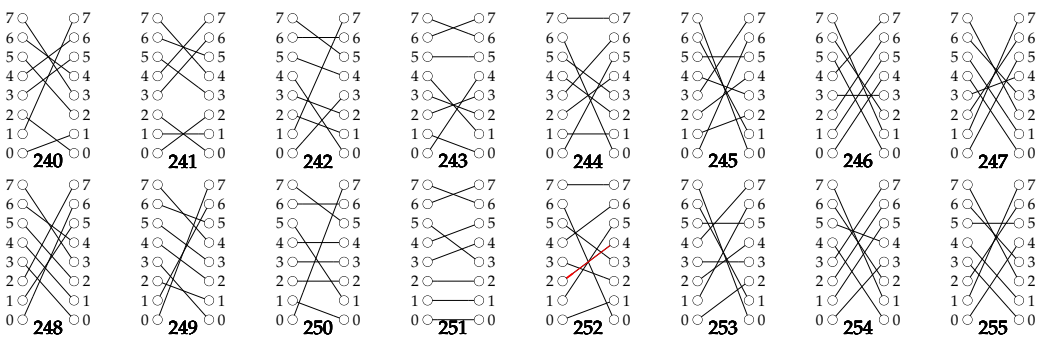
Example 9.1. The binary one-time pad of length n is the block cipher of block size n and key length n in which $e_k(x) = x + k$ for all $k \in \mathbb{F}_2^n$. For example, $e_{(1,1,1,1)}\big((1,0,0,1)\big) = (0,1,1,0)$.

Since we work over $\mathbb{F}_2$, addition is **always modulo** 2. The one-time pad has perfect secrecy (see Question 2 on Sheet 2). But it is not a good block cipher because the key can be deduced from a known plaintext/ciphertext pair (x, y) by adding x and y, to get $x + (x + k) = k$.

Modern block ciphers aim to be secure even against a chosen plaintext attack allowing *arbitrarily many* plaintexts. That is, even given all pairs $\big(x, e_k(x)\big)$ for $x \in \mathbb{F}_2^n$, there should be no faster way to find the key k then exhausting over all possible keys in the keyspace $\mathbb{F}_2^\ell$.

The following example aims to give some idea of the 'needle in haystack' effect of a strong block cipher, and why it is non-trivial to design one.

Example 9.2. Take $n = 3$ so $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^3$. The *toy block cipher* has $\mathcal{K} = \mathbb{F}_2^8$. The encryption functions are 256 of the bijections $\mathbb{F}_2^3 \to \mathbb{F}_2^3$, chosen according to a fairly arbitrary rule (details omitted). For example, the red edge in diagram **252** shows that $e_{11111100}(010) = 100$, or in decimal, $e_{\mathbf{252}}(2) = 4$



The other 240 bijections are posted on Moodle and will be available in the Q&A session in Week 9.

Suppose Alice and Bob use the toy block cipher with their shared secret key k.

(i) By a chosen plaintext attack Mark learns that $e_k(000) = 011$ and $e_k(100) = 000$. One possible key is **254**, or 11111110 in binary. There are twelve others: find at least one of them.

(ii) By choosing two further plaintexts Mark learns that $e_k(001) = 101$ and $e_k(110) = 111$. Determine k.

(iii) Later Mark's boss Eve observes the ciphertext 100. What is $d_k(100)$?

In this small example we used 256 of the $(2^3)! = 8! = 40320$ bijections of $\mathbb{F}_2^3$.

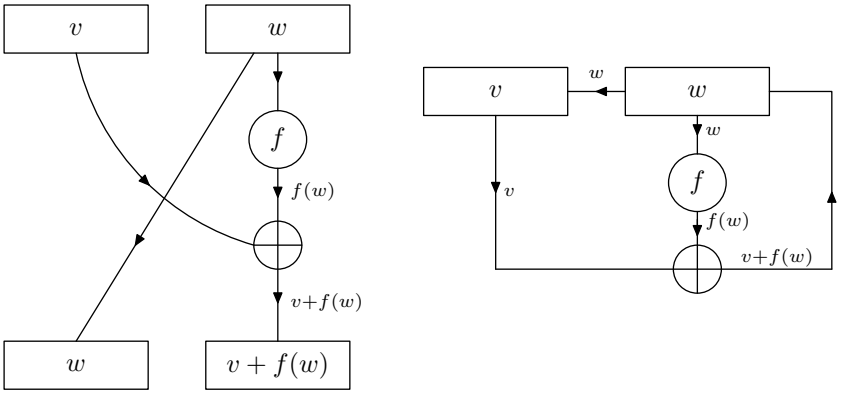
A block cipher such as AES, with block size 128 and key length 128 uses $2^{128} \approx 3.40 \times 10^{30}$ bijections of $\mathbb{F}_2^{128}$. To store *just one* of these bijections needs a list of 2^{128} pairs $(x, e_k(x))$, one pair for each $x \in \mathbb{F}_2^{128}$. Since 2^{128} bits is about 4.25×10^{28} GB, this is impractical. Instead each encryption function e_k must be computed as it is used.[16]

Feistel networks. It will be useful to represent the elements of $\mathbb{F}_2^{2m}$ as pairs (v, w) where $v, w \in \mathbb{F}_2^m$. For example $\big((1,1,1,0),(1,1,0,1)\big)$, or more briefly 1110 1101, both represent $(1,1,1,0,1,1,0,1)$.

Definition 9.3. Let $m \in \mathbb{N}$ and let $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$ be a function. The *Feistel network* for f is the function $F : \mathbb{F}_2^{2m} \to \mathbb{F}_2^{2m}$ defined by

$$F\big((v, w)\big) = \big(w, v + f(w)\big).$$

This can be compared with an LFSR: we shift (v, w) left by m positions to move w to the start. The analogue of the feedback function is $(v, w) \mapsto v + f(w)$. It is linear in v, like an LFSR, but typically non-linear in w.



[16] Computers work in binary so elements of $\mathbb{F}_2^n$ can be stored very compactly and easily manipulated. For example, key addition by $+$ in $\mathbb{F}_2^n$ corresponds to XOR, which is a single instruction: on an Intel microprocessor, if register `eax` contains $95 = 01010101$ then `xorl $15, %eax` computes $01010101 + 00001111 = 01011010$. For more see `repl.it/@mwildon/CMinimalBlockCipher`: download and then compile with `gcc -S`, or paste the C code into `godbolt.org` (compiler explorer) to see the assembly language instructions.

The circuit diagrams on the previous page show two equivalent definitions of the Feistel network: the right-hand diagram makes the analogy with LFSRs more obvious; the left-hand diagram is useful for organizing calculations (see Exercise 9.6).

Exercise 9.4. Show that, for any function $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$, the Feistel network F for f is invertible and that its inverse is $(v', w') \mapsto (w' + f(v'), v')$.

See Question 2 on Problem Sheet 6 for an extension of this exercise, showing that decryption can be performed by the same circuitry as encryption.

A block cipher of Feistel type is defined by iterating a Feistel network for a fixed number of *rounds*. The function f for each round depends on a *round key*, constructed using the key $k \in \mathbb{F}_2^\ell$.

Example 9.5 (Q-block cipher)**.** Take $m = 4$ and let

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

We define a block cipher with block size 8 and key length 12 composed of three Feistel functions. If the key is $k \in \mathbb{F}_2^{12}$ we define the three round keys by

$$k^{(1)} = (k_0, k_1, k_2, k_3), k^{(2)} = (k_4, k_5, k_6, k_7), k^{(3)} = (k_8, k_9, k_{10}, k_{11}).$$

The Feistel function in round i is $x \mapsto S(x + k^{(i)})$.

Since in each round the new left register is the old right register, we can consistently denote the output of round i by $(v^{(i)}, v^{(i+1)})$. Thus the plaintext $(v, w) \in \mathbb{F}_2^{16}$ is encrypted to the cipher text $e_k\big((v, w)\big) = (v^{(3)}, v^{(4)})$ in three rounds:

$$\begin{aligned}(v, w) = (v^{(0)}, v^{(1)}) &\mapsto \big(v^{(1)}, v^{(0)} + S(v^{(1)} + k^{(1)})\big) = (v^{(1)}, v^{(2)}) \\ &\mapsto \big(v^{(2)}, v^{(1)} + S(v^{(2)} + k^{(2)})\big) = (v^{(2)}, v^{(3)}) \\ &\mapsto \big(v^{(3)}, v^{(2)} + S(v^{(3)} + k^{(3)})\big) = (v^{(3)}, v^{(4)}).\end{aligned}$$

Exercise 9.6.

(a) Suppose that $k = 0001\,0011\,0111$, shown split into the three round keys. Show that $e_k(0000\,0000) = 1110\,0010$ and $(v^{(1)}, v^{(2)}) = (0000\,0100)$ and $(v^{(2)}, v^{(3)}) = (0100\,1110)$.

(b) Let $k' = 0001\,0011\,0000$. Show that $d_{k'}(1110\,0010) = 1100\,1100$ and $(v^{(1)}, v^{(2)}) = 1100\,1011$, $(v^{(2)}, v^{(3)}) = 1011\,1110$. Remember to use the round keys in reverse order!

(c) Suppose Eve observes the ciphertext $(v^{(3)}, v^{(4)})$ from the Q-block cipher. What does she need to know to determine $v^{(2)}$?

DES (Data Encryption Standard 1975). DES is a Feistel block cipher of block size 64. The key length is 56, so the keyspace is $\mathbb{F}_2^{56}$. Each round key is in $\mathbb{F}_2^{48}$. There are 16 rounds. (Details of how the 16 round keys are derived from the key are omitted.)

The Feistel function $f : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ is defined in three steps using eight functions $S_1, \ldots, S_8 : \mathbb{F}_2^6 \to \mathbb{F}_2^4$. Start with $x \in \mathbb{F}_2^{32}$ and a round key $k^{(i)} \in \mathbb{F}_2^{48}$. Then

(a) Expand x by a linear function (details omitted) to $x' \in \mathbb{F}_2^{48}$.
(b) Add the 48-bit round key to get $x' + k^{(i)}$.
(c) Let $x' + k^{(i)} = (y^{(1)}, \ldots, y^{(8)})$ where $y^{(j)} \in \mathbb{F}_2^6$ for each j. Let

$$z = \big(S_1(y^{(1)}), \ldots, S_8(y^{(8)})\big) \in \mathbb{F}_2^{32}.$$

(d) Apply a bijection (details omitted) of the positions of z.

Note that (a) and (d) are linear, and (b) is a conventional key addition in $\mathbb{F}_2^{48}$. So the *S-boxes* in (c) are the only source of non-linearity. (Here 'S' stands for 'substitution'.)

- The aim of (c) is 'confusion': to make the relationship between nearby bits of the plaintext and ciphertext complicated and non-linear.
- The aim of (d) is 'diffusion': to turn confusion between nearby bits into long range confusion.

In 1994 a sub-exhaustive attack on DES using linear cryptanalysis (part of the M.Sc. course) on 2^{48} known plaintext/ciphertext pairs was discovered.

At about the same time, the relatively small keyspace $\mathbb{F}_2^{56}$ meant that exhaustive attacks became practical. Therefore DES cannot be considered secure. Some timings for exhaustive attacks:

- 1997: 140 days, distributed search on internet
- 1998: 9 days 'DES cracker' special purpose $250000
- 2017: 1 day 'COPACOBANA' 35 FPGAs $10000 (estimate)

Exercise 9.7. Let 2DES be the block cipher defined by applying DES twice, first with key $k \in \mathbb{F}_2^{56}$ then with $k' \in \mathbb{F}_2^{56}$. So the keyspace is $\mathbb{F}_2^{56} \times \mathbb{F}_2^{56}$ and for $(k, k') \in \mathbb{F}_2^{56} \times \mathbb{F}_2^{56}$,

$$e_{(k,k')}(x) = e'_k\big(e_k(x)\big).$$

How long would a brute force exhaustive search over $\mathbb{F}_2^{56} \times \mathbb{F}_2^{56}$ take?

In the following attack, blue letters and $\star$ denotes known quantities the attacker chooses or can obtain by encrypting using her black box. Red denotes unknown quantities. Using colours is not a standard convention, and you are welcome to ignore it if you prefer.

Attack 9.8. *We attack 2DES supposing, in a chosen plaintext attack, that we can encrypt arbitrary plaintexts. Let (k, k') be the secret key. We choose $x \in \mathbb{F}_2^{64}$ and, using the black box in our possession, find $e_{(k,k')}(x) = z \in \mathbb{F}_2^{64}$.*

(1) *For every $k_\star \in \mathbb{F}_2^{56}$ we calculate $e_{k_\star}(x)$, and separately, for every $k'_\star \in \mathbb{F}_2^{56}$, we calculate $d_{k'_\star}(z)$. After 2^{56} encryptions and 2^{56} decryptions we have the sets*

$$E = \{(k_\star, e_{k_\star}(x)) : k_\star \in \mathbb{F}_2^{56}\}$$
$$D = \{(k'_\star, d_{k'_\star}(z)) : k'_\star \in \mathbb{F}_2^{56}\}.$$

(2) *Using these sets we calculate, for each $y_\star \in \mathbb{F}_2^{56}$, those key pairs (k, k') that 'meet-in-the-middle' at $y_\star$:*

$$K_{y_\star} = \{(k_\star, k'_\star) : e_{k_\star}(x) = d_{k'_\star}(z)\}.$$

Setting $y = e_k(x) = d_{k'}(z)$, we have $(k, k') \in K_y$. So the correct key is in one of the sets $K_{y_\star}$.

(3) *To find (k, k'), we choose another plaintext X. Then for each $y_\star \in \mathbb{F}_2^{56}$ and each $(k_\star, k'_\star) \in K_{y_\star}$ we test if*

$$e_{(k_\star,k'_\star)}(X) = e_{(k,k')}(X).$$

It is very likely that only the correct key passes this test.

Exercise 9.9. The total number of key pairs $(k_\star, k'_\star) \in E \times D$ we test in (3) is $M = \sum_{y \in \mathbb{F}_2^{64}} |K_y|$. Assuming that the encryption function in DES behave like random bijections of $\mathbb{F}_2^{64}$, show that the expected size of each K_y is $\left(2^{56}/2^{64}\right)^2 = \frac{1}{2^{16}}$, and so the expected size of M is $2^{64} \times \frac{1}{2^{16}} = 2^{48}$. Why is it then very likely that only the correct key passes the test that $e_{(k_\star,k'_\star)}(X) = e_{(k,k')}(X)$? Deduce that Attack 9.8 uses about $2^{56} + 2^{56} + 2 \times 2^{48}$ encryptions/decryptions. Does this make the attack subexhaustive?

See Question 4 on Problem Sheet 7 for 3DES (Triple-DES): it has keyspace $\mathbb{F}_2^{56} \times \mathbb{F}_2^{56} \times \mathbb{F}_2^{56}$ and encryption functions defined by

$$e_{(k,k',k'')}(x) = e''_k\left(d'_k(e_k(x))\right).$$

The DES model, of combining a non-linear S-box with linear maps and key additions in $\mathbb{F}_2^n$, is typical of block ciphers.

Modes of operation. A block cipher of block size n encrypts plaintexts in $\mathbb{F}_2^n$ to ciphertexts in $\mathbb{F}_2^n$. If the message x is longer than n bits, it must be split into blocks $x^{(1)}, \ldots, x^{(m)} \in \mathbb{F}_2^n$ suitable for encryption:

$$x = (x^{(1)}, \ldots, x^{(m)}).$$

Fix a key $k \in \mathcal{K}$: this is only key used.

- In Electronic Codebook Mode, the encryption function e_k is applied to each block in turn:

$$x^{(1)} \mapsto e_k(x^{(1)}), x^{(2)} \mapsto e_k(x^{(2)}), \ldots, x^{(m)} \mapsto e_k(x^{(m)})$$

- Cipher Block Chaining:

$$\begin{aligned} x^{(1)} &\mapsto e_k(x^{(1)}) = y^{(1)} \\ x^{(2)} &\mapsto e_k(y^{(1)} + x^{(2)}) = y^{(2)} \\ &\vdots \\ x^{(m)} &\mapsto e_k(y^{(m-1)} + x^{(m)}) = y^{(m)} \end{aligned}$$

If $x^{(i)} = x^{(j)}$ then, in Electronic Codebook Mode, the ciphertext blocks $e_k(x^{(i)})$ and $e_k(x^{(j)})$ are equal. This leads to frequency attacks, as seen in Example 2.5 for the substitution cipher. This is a weakness of the mode of operation, not of the underlying block cipher. Cipher Block Chaining avoids this problem.

10. Differential cryptanalysis and AES

Recall that in a *chosen plaintext* attack you get to pick a plaintext x and are given $e_k(x)$, and in a *chosen ciphertext* attack you get to pick a ciphertext y and are given $d_k(y)$.

Question. How can modern block ciphers be attacked if we are allowed arbitrarily many encryptions and decryptions?

Differential cryptanalysis was known to the designers of DES in 1974; the S-boxes were chosen to resist the difference attack below. They kept this attack secret, at the request of the NSA.

One important idea is seen in the attack on the reused one-time pad in Question 4 on Problem Sheet 3. We have unknown plaintexts x, $x_{\mathbf{\Delta}} \in \mathbb{F}_2^n$, an unknown key $k_{\text{otp}} \in \mathbb{F}_2^n$, and known ciphertexts $x + k_{\text{otp}}$ and $x_{\mathbf{\Delta}} + k_{\text{otp}}$. Adding the known ciphertexts gives $x + x_{\mathbf{\Delta}}$, independent of k_{otp}.

Put another way, if two plaintexts x, x_Δ differ by a *difference* $\mathbf{\Delta}$, so $x + x_\Delta = \mathbf{\Delta}$, then so do their encryptions: $(x + k_{\text{otp}}) + (x_\Delta + k_{\text{otp}}) = \mathbf{\Delta}$.

As a notational aid, we write differences in bold in the printed notes (but not on the board): again this is optional and non-standard.

Attack on the Q-block cipher. Recall that we may write elements as $\mathbb{F}_2^8$ as pairs (v, w) where $v \in \mathbb{F}_2^4$ and $w \in \mathbb{F}_2^4$. In round 1 of the Q-block cipher (see Example 9.5), the Feistel network sends (v, w) to $\big(w, v + S(w + k^{(1)})\big)$ where

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

Lemma 10.1.

(i) *For any $w \in \mathbb{F}_2^4$ we have $S(w + \mathbf{1000}) = S(w) + \mathbf{0010}$.*

(ii) *For any $(v, w) \in \mathbb{F}_2^8$ and any round key $k^{(1)} \in \mathbb{F}_2^4$ round 1 of the Q-block cipher is*

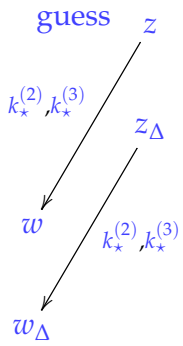
$$\big(v + \mathbf{0000}, w + \mathbf{1000}\big) \mapsto \big(w, v + S(w + k^{(1)})\big) + \mathbf{1000\,0010}.$$

Thus the first round of the Q-block cipher encrypts plaintexts differing by **0000 1000** to intermediate ciphertexts differing by **1000 0010**. *This 'deterministic' behaviour is just like the one-time pad.* This makes the Q-block cipher vulnerable to a difference attack using chosen plaintexts and ciphertexts.

Example 10.2. Let $x \in \mathbb{F}_2^8$ and let $\boldsymbol{\Delta} = \mathbf{0000\,1000} \in \mathbb{F}_2^8$. The diagram below shows the encryption of x and $x_\Delta = x + \boldsymbol{\Delta}$ over the three rounds of the Q-block cipher using the key $k = (k^{(1)}, k^{(2)}, k^{(3)})$, split into three round keys:

$$\begin{array}{ccccc} x & \xrightarrow{k^{(1)}} & y & \xrightarrow{k^{(2)},\, k^{(3)}} & z \\ \boldsymbol{\Delta} = \mathbf{0000\,1000} & & \boldsymbol{\Delta}' = \mathbf{1000\,0010} & & \boldsymbol{\Gamma} \\ x_\Delta & \xrightarrow{k^{(1)}} & y_\Delta & \xrightarrow{k^{(2)},\, k^{(3)}} & z_\Delta \end{array}$$

The middle differences are $\boldsymbol{\Delta} = x + x_\Delta$ and $\boldsymbol{\Delta}' = y + y_\Delta$. We know $\boldsymbol{\Delta}'$ by Lemma 10.1(ii).



We attack by guessing $k_\star^{(2)}$ and $k_\star^{(3)}$. We use these guesses to decrypt the ciphertexts z and z_Δ **over two rounds**, obtaining the intermediate ciphertexts w and w_Δ. On a correct guess $k_\star^{(2)} = k^{(2)}$ and $k_\star^{(3)} = k^{(3)}$ and then $w = y$ and $w_\Delta = y_\Delta$ and $w + w_\Delta = \boldsymbol{\Delta}'$.

To test our guess we compute the difference $\boldsymbol{\Delta}_\star = w + w_\Delta$. If $\boldsymbol{\Delta}_\star \neq \boldsymbol{\Delta}'$, we know our guess is wrong.

To see this in practice, take $k = 0001\,0011\,0111$ and $x = 0000\,0000$. (For this example, we have chosen k, but from the attacker's perspective, it is unknown.) By Exercise 9.6(i), $z = 1110\,0010$; a similar calculation gives $z_\Delta = 1101\,1100$.

(1) If we guess that $k^{(2)} = 0011$, $k^{(3)} = 0000$ then $w = 1100\,1011$, as can be read from $(v^{(1)}, v^{(2)})$ in Example 9.6(ii), and $w_\Delta = 1111\,1011$. Hence $\boldsymbol{\Delta}_\star = \mathbf{0011\,0000}$ and we know this guess is wrong.

(2) If we guess that $k^{(2)} = 0001$, $k^{(3)} = 1111$ then $w = 0000\,0110$ and $w_\Delta = 1000\,0100$. Hence $\boldsymbol{\Delta}_\star = \mathbf{1000\,0010}$ and we do not know that the guess is wrong. (This example was chosen so that also $w_0w_1w_2w_3 = x_4x_5x_6x_7$, as required by the Feistel function.)

In total there are 16 pairs $(k_\star^{(2)}, k_\star^{(3)}) \in \mathbb{F}_2^8$ such that $\boldsymbol{\Delta}_\star = \boldsymbol{\Delta}'$, namely all binary words of the form $\star\star\star1\star1bb$ where $b \in \{0,1\}$. Trying each guess together with all 16 possibilities for $k_\star^{(1)} \in F_2^4$ and comparing the encryption of x using $k_\star^{(1)}, k_\star^{(2)}, k_\star^{(3)}$ with z, shows that

$$k \in \{0001\,0011\,0111,\ 0010\,1111\,0100,\ 1001\,0001\,1111,\ 1010\,1101\,1100\}.$$

All these keys encrypt x to z and x_Δ to z_Δ. Repeating the attack with a different plaintext shows that k is either the first or third key.

That we are left with two keys is explained by Question 2(c) on Sheet 7: it follows from Lemma 10.1(i) that, in the Q-block cipher, the encryption functions e_k and $e_{k+1000\,0010\,1000}$ are the same.

Exercise 10.3. Assume that the difference attack shows the key is one of 16 possible $(k_\star^{(2)}, k_\star^{(3)})$. Show that it is subexhaustive: that is, it requires less computing than trying all $2^{12} = 4096$ keys.

AES (Advanced Encryption Standard 2002). AES is the winner of an open competition to design a successor to DES. Its block size is 128 and its key length is 128 (with variants allowing 192 and 256). It is not a Feistel cipher, but it is still built out of multiple rounds, like DES. It is the most widely used block cipher. No-one has found a subexhaustive attack on AES, despite the huge incentive. The remaining material below is 'extra', and included for interest only.

Extra: the definition of AES. The two main building blocks in AES are shown in the following two examples.

Example 10.4. The *affine block cipher* of block size n has keyspace all pairs (A, b), where A is an invertible $n \times n$ matrix with entries in $\mathbb{F}_2$ and $b \in \mathbb{F}_2^n$. The encryption functions $e_{(A,b)} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ are the *affine transformations* defined by

$$e_{(A,b)}(x) = xA + b.$$

As an exercise, show that the decryption functions are $d_{(A,b)}(y) = (y - b)A^{-1}$. The key in the affine block cipher can be deduced by a known plaintext attack using $n + 1$ chosen plaintexts: for example, encrypting $0 \in \mathbb{F}_2^n$ reveals b.

The affine block cipher gives excellent diffusion but poor confusion.

Definition 10.5. Let z be an indeterminate, as used in the **M.Sc.** course for polynomials and power series. Define

$$\mathbb{F}_{2^8} = \{x_0 + x_1 z + \cdots + x_7 z^7 : x_0, x_1, \ldots, x_7 \in \mathbb{F}_2\}.$$

Elements of $\mathbb{F}_2^8$ are added and multiplied like polynomials in z, but whenever you see a power z^d where $d \geq 8$, eliminate it using the rule

$$z^8 = 1 + z + z^3 + z^4.$$

For example $(1 + z) + (z + z^5) = 1 + z^5$ and

$$z^9 = z \times z^8 = z(1 + z + z^3 + z^4) = z^2 + z^3 + z^4 + z^5.$$

Multiplying the defining rule for z by z^{-1}, we get $z^{-1} + 1 + z^2 + z^3 + z^7 = 0$ so $z^{-1} = 1 + z^2 + z^3 + z^7$. In fact the strange rule[17] for eliminating

[17] An equivalent definition using ring theory is $\mathbb{F}_{2^8} = \mathbb{F}_2[z]/\langle 1 + z + z^3 + z^4 + z^8\rangle$; now z is the coset $z + \langle 1 + z + z^3 + z^4 + z^8\rangle$ in the quotient ring. In the quotient ring, $z^8 + \langle 1 + z + z^3 + z^4 + z^8\rangle = 1 + z + z^3 + z^4 + \langle 1 + z + z^3 + z^4 + z^8\rangle$, justifying the rule for eliminating z^8. The polynomial $1 + z + z^3 + z^4 + z^8$ was chosen by the designers of AES: it is irreducible (hence it generates a maximal ideal, and the quotient of $\mathbb{F}_2[z]$ by this ideal is a field) but not primitive.

powers of z^8 (and higher powers) means that every non-zero element of $\mathbb{F}_{2^8}$ has a multiplicative inverse. Hence $\mathbb{F}_{2^8}$ is a *field*. For example

$$(1+z^2)^{-1} = z + z^4 + z^6,$$

as you can check by multiplying $(1+z^2)(z+z^4+z^6)$ and eliminating z^8.

Definition 10.6. Define $s : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ by

$$s(\beta) = \begin{cases} \beta^{-1} & \text{if } \beta \neq 0 \\ 0 & \text{if } \beta = 0. \end{cases}.$$

Let $S : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ be the corresponding function defined by identifying $\mathbb{F}_2^8$ with $\mathbb{F}_{2^8}$ by $(x_0, x_1, \ldots, x_7) \longleftrightarrow x_0 + x_1 z + x_2 z^2 + \cdots + x_7 z^7$.

The MATHEMATICA notebook `BlockCiphers.nb` includes an implementation of s and S, and can be used to calculate in $\mathbb{F}_{2^8}$.

Example 10.7. Writing elements of $\mathbb{F}_2^8$ as words of length 8 (with a small space for readability):

(1) $1000\,0000 \longleftrightarrow 1 \in \mathbb{F}_{2^8}$ and $1^{-1} = 1$, so $s(1) = 1$ and $S(1000\,0000) = 10000000$;

(2) $0100\,0000 \longleftrightarrow z \in \mathbb{F}_{2^8}$ and $z^{-1} = 1 + z^2 + z^3 + z^7$ was seen above, so $s(z) = 1 + z^2 + z^3 + z^7$ and $S(0100\,0000) = 10110001$.

(3) *Exercise:* Find $s(z^2)$ and hence show $S(0010\,0000) = 1101\,0011$.

Definition of AES. There are 10 rounds in AES. In each round, the input $x \in \mathbb{F}_2^{128}$ is split into $128/8 = 16$ subblocks each in $\mathbb{F}_2^8$.

- The round key in $\mathbb{F}_2^{128}$ is added (ADDROUNDKEY).
- The pseudo inverse function $S : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ is applied to each subblock *followed* by an affine transformation $\mathbb{F}_2^8 \to \mathbb{F}_2^8$, of the type in Example 10.4. This gives confusion and diffusion *within each subblock*. (SUBBYTES.)
- Diffusion across all 128 bits comes from a row bijection of the 16 subblocks, organized into a 4×4 grid

$$\begin{array}{cccc} q(0) & q(4) & q(8) & q(12) \\ q(1) & q(5) & q(9) & q(13) \\ q(2) & q(6) & q(10) & q(14) \\ q(3) & q(7) & q(11) & q(15) \end{array} \longrightarrow \begin{array}{cccc} q(0) & q(4) & q(8) & q(12) \\ q(13) & q(1) & q(5) & q(9) \\ q(10) & q(14) & q(2) & q(6) \\ q(7) & q(11) & q(15) & q(3) \end{array}$$

  and a further mixing of each column by the affine block cipher (SHIFTROWS and MIXCOLUMNS)

AES was defined to be efficient in hardware: for example, the subblocks fit exactly into 8-bit bytes. Encryption and decrption are single instruction operands on modern Intel and AMD microprocessors. In practice AES is about six times faster than 3DES.

There are also versions of AES defined with keyspace $\mathbb{F}_2^{192}$ and $\mathbb{F}_2^{256}$, using 12 or 14 rounds, respectively.

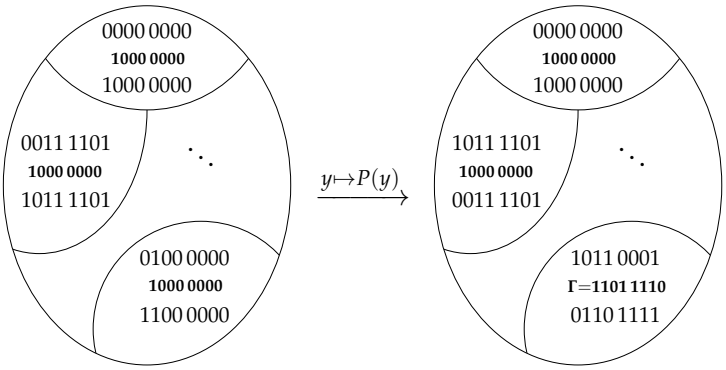
Pseudo-inversion resists the difference attack.

Lemma 10.8. *Let $\gamma \in \mathbb{F}_2^8$ be non-zero. Then*

$$\{\beta \in \mathbb{F}_{2^8} : s(\beta) + s(\beta + 1) = \gamma\}$$

has size 0 or 2, except when $\gamma = 1$, when it is $\{0, 1, \zeta, 1 + \zeta\}$ where $\zeta = z^2 + z^3 + z^4 + z^5 + z^7$.

The analogous result holds for $S : \mathbb{F}_2^8 \to \mathbb{F}_2^8$. It is illustrated by the diagram below.



Let $\mathbf{\Delta} = 1000\,0000$, corresponding to $1 \in \mathbb{F}_{2^8}$. The left diagram shows $\mathbb{F}_2^8$ partitioned into pairs $\{x, x_\Delta\}$ with $x + x_\Delta = \mathbf{\Delta}$. The *output difference* $S(x) + S(x_\Delta)$ can be any of 127 elements $\mathbf{\Gamma} \in \mathbb{F}_2^8$. Unless $\mathbf{\Gamma} = \mathbf{1000\,0000}$, the pair $\{x, x_\Delta\}$ for output difference $\mathbf{\Gamma}$ is unique (as in the bottom-right of the diagram). Exceptionally, when $\mathbf{\Gamma} = \mathbf{1000\,0000}$, there are two possible pairs (shown in the top-left of the diagram).

Exercise 10.9. Explain why the output difference cannot be **0000 0000**.

Suppose we encrypt two plaintexts x, $x_\Delta \in \mathbb{F}_2^{128}$ differing by $\mathbf{\Delta}$ using one round of AES. In the first step of the first round, an unknown round key k_{round} is added, to give $x + k_{\text{round}}$ and $x_\Delta + k_{\text{round}}$. The difference is still Δ. But by Lemma 10.8, there are 127 (almost) equally likely output differences $\mathbf{\Gamma}$. The difference attack is ineffective.

Extra: Physical limits to computation. In *Computational capacity of the universe*, Seth Lloyd, *Phys. Rev. Lett.* (2002) **88** 237901, the author argues that even if the universe is one vast computer[18] then it cannot have performed more than 10^{120} operations. Assuming one encryption per operation, this would exhaust the keyspace of a block cipher with key length 398. Similar arguments suggest that a block cipher of key length 128 with no subexhaustive attacks, such as AES, is in practice secure.

[18]The temptation to cite Douglas Adams must have been hard to resist.

(D) Public key ciphers and digital signatures

11. Introduction to public key cryptography

So far in this course we have seen symmetric ciphers, which require Alice and Bob to exchange the secret key before they communicate securely. You may have noticed this does not appear to be necessary when you use the internet.

> **Question.** How can Alice and Bob establish a shared secret key communicating only over the insecure channel on page 4?

In this part, everything in red is intended to be private. Everything not in red is known to the whole world— this includes the eavesdropper Eve. (You are welcome to ignore this non-standard convention if you prefer.)

Example 11.1. Alice and Bob need a 128-bit key for use in AES. They agree a prime p such that $p > 2^{128}$. Then

(1) Alice chooses a secret $a \in \mathbb{N}$ with $1 \leq a < p$. Bob chooses a secret $b \in \mathbb{N}$ with $1 \leq b < p$.
(2) Alice sends Bob $2^a \bmod p$. Bob sends Alice $2^b \bmod p$. (Note that a and b are secret, but $2^a \bmod p$ and $2^b \bmod p$ are sent publically.)
(3) Alice computes $(2^b \bmod p)^a \bmod p$ and Bob computes $(2^a \bmod p)^b \bmod p$.
(4) Now Alice and Bob both know $2^{ab} \bmod p$. They each calculate the number $2^{ab} \bmod p$ in binary and take its final 128 bits to get an AES key.

After (2), the eavesdropper Eve knows p, $2^a \bmod p$ and $2^b \bmod p$. It is believed that it is hard for her to use this information to find $2^{ab} \bmod p$. The difficulty can be seen even in small examples.

After (4) Alice and Bob can communicate using the AES cryptosystems, which has no known sub-exhaustive attacks.

So remarkably, Alice and Bob can communicate securely *without exchanging any private key material*.

Exercise 11.2. Let $p = 11$. As Eve you know that Alice has sent Bob 6. Do you have any better way to find a such that $2^a = 6$ than trying each possibility?

n	0	1	2	3	4	5	6	7	8	9
$2^n \bmod 11$	1	2	4	8	5	10	9	7	3	6

To compute this table it is not necessary to calculate, for instance, $2^8 = 256$, and then reduce it modulo 11. Instead, just double the previous entry. Thus from $2^7 \equiv 7 \bmod 11$ we get $2^8 \equiv 7 \times 2 = 14 \equiv 3 \bmod 11$. Similarly, in

$$(2^a \bmod p)^b \bmod p = 2^{ab} \bmod p = (2^b \bmod p)^a \bmod p$$

step (4) of Example 11.1 we always work with numbers mod p.

This exercise shows two further number-theoretic facts that will be needed below. (See also Fact 11.4 below.)

- Fermat's Little Theorem: $c^{p-1} \equiv 1 \bmod p$ for any c not divisible by p.
- If $g^m \not\equiv 1 \bmod p$ for all m such that $1 \leq m < p-1$, then g is said to be a *primitive root modulo* p. If g is a primitive root then, working modulo p, we have

  $$\{1, g, g^2, \ldots, g^{p-2}\} = \{1, 2, \ldots, p-1\}$$

  Primitive roots always exist[19]: in Exercise 11.2 we took $g = 2$.

Note that 2 is not always a primitive root: for example if $p = 127$ then we have $2^7 = 128 \equiv 1 \bmod 127$, so the powers of 2 are $\{1, 2, 4, 8, 16, 32, 64\}$, giving only 7 of the 126 non-zero elements.

Diffie–Hellman key exchange. This is nothing more than Example 11.1, modified to avoid some potential weaknesses, and implemented efficiently. The protocol is still not secure against a Man-in-middle attack, but this can be fixed: see the extras for §12.

- The prime p is chosen so that $p-1$ has at least one large prime factor. (This is true of most primes. There are fast ways to decide if a number is prime.)
- Rather than use 2, Alice and Bob use a primitive root modulo p, so every element of $\{1, \ldots, p-1\}$ is congruent to a power of g. (The base is public.)

[19] Let $\mathbb{Z}_p^\times = \{1, \ldots, p-1\}$ be the multiplicative group of $\mathbb{Z}_p$. *Claim:* $\mathbb{Z}_p^\times$ is cyclic of order $p-1$. *Proof:* let t be the lowest common multiple of the orders of all the elements of $\mathbb{Z}_p^\times$. Then $x^t = 1$ for all $x \in \mathbb{Z}_p^\times$. But a polynomial of degree t has at most t roots, hence $t \geq p-1$. By the lemma below, there is an element g of order t. By Fermat's Little Theorem (or Lagrange's Theorem is you prefer), the order of g is at most $p-1$. □

Lemma: if an abelian group A has elements g and g' of order d and d' respectively, then it has an element of order $\mathrm{lcm}(d, d')$.

Proof of lemma: let $d = p^e c$ and $d' = p^{e'} c'$ with c, c' coprime to the chosen prime p, and assume without loss of generality that $e \geq e'$. By induction A has an element h of order $\mathrm{lcm}(c, c')$; now $g^c h$ is the product of elements of coprime orders p^e and $\mathrm{lcm}(c, c')$ in an abelian group, so has order $p^e \mathrm{lcm}(c, c') = \mathrm{lcm}(d, d')$.

- Alice and Bob compute $g^a \bmod p$ and $g^b \bmod p$ by repeated squaring: see Question 1 on Sheet 9. This method is faster than the repeated doubling seen in Exercise 11.2. Either method shows that g^a can be computed using only numbers of size about p.
- The shared key is $g^{ab} \bmod p$.

Diffie–Hellman can be turned into the ElGamal cryptosystem: see Question 6 on Sheet 9. But it is faster to use it, as defined above, to establish a shared key, and then use this key with a fast block cipher such as AES.

One-way functions. A *one-way function* is a bijective function that is fast to compute, but whose inverse is hard to compute. It is beyond the scope of this course to make this more precise.

It is not known whether one-way functions exist. Their existence implies P $\neq$ NP: very roughly, if P = NP then any problem whose solution is quick to check, such as Sudoku, is also quick to solve. It is widely believed that P $\neq$ NP, but no proof is known.

Diffie–Hellman key exchange is secure only if, given g and $g^x \bmod p$, it is hard to find x. (This is called the Discrete Log Problem.) Equivalently, the function

$$f : \{0, \ldots, p-2\} \to \{1, \ldots, p-1\}$$

defined by $f(x) = g^x \bmod p$, is one-way. This is widely believed to be the case. But it more likely that the Discrete Log Problem is easy than that AES has a sub-exhaustive attack.

Inverting modular exponentiation. In the RSA cryptosystem, we use modular exponentiation as the encryption map. We therefore need to know when it is invertible.

Lemma 11.3. *If p is prime and $\mathrm{hcf}(a, p-1) = 1$ then the inverse of $x \mapsto x^a$ mod p is $y \mapsto y^r$ mod p, where $ar \equiv 1$ mod $p-1$.*

For example, $x \mapsto x^3 \bmod 29$ is invertible, with inverse $y \mapsto y^{19} \bmod 29$. This works, since after applying both functions, in either order, we send x to x^{57}; by Fermat's Little Theorem, $x^{57} = x^{28\times 2+1} = (x^{28})^2 x \equiv x \bmod 29$. On the other hand $x \mapsto x^7 \bmod 29$ is not invertible: working mod 29 the image is $\{1, 2^7, 2^{14}, 2^{21}\} = \{1, 12, 28, 17\}$.

Given p and a with $\mathrm{hcf}(a, p-1) = 1$, one can use Euclid's algorithm to find $s, t \in \mathbb{Z}$ such that $as + (p-1)t = 1$. Then $as = 1 - pt$ so $as \equiv 1 \bmod p-1$, and we take $r \equiv s \bmod p-1$. For example, if $p = 29$ and $a = 5$ then we have $28 = 9 \times 3 + 1$ so

$$1 = 3 \times (-9) + 28 \times 1$$

and $s = -9$. Since $-9 \equiv 19 \bmod 28$, we take $r = 19$, as above.

This example shows all the ideas needed for the proof of Lemma 11.3, and shows that it is fast to find r. Thus we cannot use $x \mapsto x^a \bmod p$ as a secure encryption function.

Fact 11.4. *Let p and q be distinct primes. Let $n = pq$. If*

$$\mathrm{hcf}\big(a, (p-1)(q-1)\big) = 1$$

then $x \mapsto x^a$ mod n is invertible with inverse $y \mapsto y^r$ mod n, where $ar \equiv 1$ mod $(p-1)(q-1)$.

Example 11.5. Let $p = 11$, $q = 17$, so $n = pq = 187$ and $(p-1)(q-1) = 160$. Let $a = 9$. Adapting the proof for Lemma 11.3, we use Euclid's Algorithm to solve $9s + 160t = 1$, getting $s = -71$ and $t = 4$. Since $-71 \equiv 89 \bmod 160$, the inverse of $x \mapsto x^9 \bmod 187$ is $y \mapsto y^{89} \bmod 187$.

Thus given p, q and a, it is easy to find r as in Fact 11.4. But it is believed to be hard to find r given only n and a. If so, $x \mapsto x^a \bmod n$ is a one-way function, suitable for use as the encryption function in a cryptosystem.

In this context the term *trapdoor function* is also used: knowing the trapdoor, here the factors p and q, makes it easy to compute the inverse.

By contrast, the function $f : \{0, \ldots, p-2\} \to \{1, \ldots, p-1\}$ defined by $f(x) = g^x$ is not a suitable encryption function, since while it is believed to be one-way, there is no known trapdoor that makes it fast to compute the inverse.

RSA Cryptosystem. Let $n = pq$ be the product of distinct primes p and q. In the RSA Cryptosystem, with *RSA modulus* n,

$$\mathcal{P} = \mathcal{C} = \{0, 1, \ldots, n-1\}$$

and

$$\mathcal{K} = \big\{(p, q, a) : a \in \{1, \ldots, n-1\}, \mathrm{hcf}\big(a, (p-1)(q-1)\big) = 1\big\}.$$

The *public key* corresponding to (p, q, a) is (n, a) and the *private key* corresponding to (p, q, a) is (n, r), where $ar \equiv 1 \bmod (p-1)(q-1)$. (Note that a is part of the public key, so unlike Diffie–Hellman, it is public.) The encryption function for (p, q, a) is

$$x \mapsto x^a \bmod n$$

and the decryption function is

$$y \mapsto y^r \bmod n.$$

Note that anyone knowing the public key can encrypt, but only someone knowing the private key, or the entire key (p, q, a) can decrypt.

Example 11.6.

(1) For a small example, take p and q as in Example 11.5. If Alice's public key is $(187, 9)$ then her private key is $(187, 89)$. If Bob's message is 10 then he sends 109 to Alice, since $10^9 \equiv 109 \bmod 187$. Alice decrypts to 10 by computing $109^{89} \bmod 187$.

(2) The MATHEMATICA notebook `PKC.nb` available from Moodle can be used when p and q are bigger.

Typically p and q are chosen so that the standard exponent $a = 2^{16} + 1 = 65537$ is coprime to $(p-1)(q-1)$. Since $2^{16} + 1$ is prime, this can be checked just by dividing $p - 1$ and $q - 1$ by $2^{16} + 1$. Then $x^a \bmod n$ can be computed quickly by repeated squaring, as in Question 1 on Problem Sheet 9.

Question 6 on Sheet 9 shows that knowing $(p-1)(q-1)$ and n is equivalent to knowing p and q; this makes it unlikely that there is an attack on RSA other than by factorizing n. The extras for this section show that given the private key (n, r) it is easy to find p and q.

The best known factoring algorithm is the Number Field Sieve. It was used to factorize a 768 bit n in 2010. This took about 1500 computer years, in 2010 technology. NIST (the US standard body) now recommend that n should have 2048 bits.

Why don't we just use Public Key Cryptography? The recommended key lengths for public key cryptosystems are typically much longer than for symmetric cryptosystems such as AES. Public key encryption is also much slower. Unlike Diffie–Hellman and RSA, block ciphers such as AES are widely believed to be resistant to quantum attacks: see the extras for this section.

Extra: some history. Diffie–Hellman Key Exchange was published[20] in 1976. The RSA Cryptosystem, named after Rivest, Shamir and Adleman was published[21] in 1977. Both papers are clearly written and worth reading—as here, the original account is often one of the best.

It emerged in 1997 that the RSA cryptoscheme had been discovered in GCHQ in 1973 by Cocks, building on work of another GCHQ-insider, Ellis, who had suggested in 1969 that 'non-secret' encryption might be possible. Later in 1973 Williamson discovered Diffie–Hellman Key Exchange. See `www.wired.com/1999/04/crypto/` for a good account.

[20] Diffie, Whitfield; Hellman, Martin E., *New directions in cryptography*, IEEE Trans. Information Theory **22** (1976) 644–654.

[21] Rivest, R. L.; Shamir, A.; Adleman, L., *A method for obtaining digital signatures and public-key cryptosystems*, Comm. ACM **21** (1978) 120–126.

Extra: factoring n given an RSA private key (n, r). Suppose we somehow learn Alice's private key (n, r). We know her public key (n, a), so can compute ar. By choice of r, we know that $ar \equiv 1 \bmod (p-1)(q-1)$. Hence $ar - 1$ is a multiple of $(p-1)(q-1)$.

Let t be obtained by dividing $ar - 1$ by small odd primes until a factor is found. There is a good chance that $p - 1$ divides t and $q - 1$ does not, or *vice versa*. Assuming the first case, Fermat's Little Theorem implies that $x^t \equiv 1 \bmod p$ for all x not divisible by p. Moreover, because $\mathrm{hcf}(t, q-1)$ is a proper factor of the order $q - 1$ of the group $\mathbb{Z}_q^\times$, $x^t \not\equiv 1 \bmod q$ for most x. Therefore for most x, p divides $x^t - 1 \bmod n$, but q does not and so computing $\mathrm{hcf}(x^t - 1, n) = p$ reveals p.

This attack is related to the Pollard ρ-factoring method, which you can learn about on the web or in our computational number theory course.

Example 11.7. As in the example in lectures we suppose Alice generates an RSA key using MATHEMATICA, defining p and q by `NextPrime[2^80]` and `NextPrime[2^81]`.[22] Alice publishes $(n, 65537)$ as her public key. Her decryption exponent r, found by `PowerMod[65537,-1,(p-1)(q-1)]` is $2\,486\,450 \ldots 629441 \approx 2.4 \times 10^{48}$.

Suppose that Mark the Mole learns her private key (n, r). He computes $ar - 1 = 162\,954 \ldots 674\,816 \approx 1.6 \times 10^{53}$. Then by trial division

```
Select[Range[1, 1000], Mod[c*r - 1, #] == 0~And~PrimeQ[#] &]
```

in MATHEMATICA, he finds that the smallest prime factors of $ar - 1$ are $2, 3, 43, 617, \ldots$. Since 9 divides $ar - 1$, it is possible that 3 divides both $p - 1$ and $q - 1$, so instead he uses 43 and takes $t = (ar - 1)/43$. Trying $x = 2$ he computes $2^t - 1 \bmod n$ using `PowerMod[2, t, n] - 1` and then $\mathrm{hcf}(2^t - 1, n)$ using `GCD[PowerMod[2, t, n] - 1, n]`. This highest common factor turns out to be p. (The numbers themselves are too big to write here.)

You can check this using the MATHEMATICA notebook on Moodle. It has a similar example with primes of the cryptographically standard size: $p, q \approx 2^{1024}$.

Why it worked: Factoring $p - 1$ and $q - 1$ shows that

$$p - 1 = 2^2 \times 1093 \times 31039 \times 8\,908\,647\,580\,887\,961$$

$$q - 1 = 2^4 \times 3 \times 43 \times 617 \times 683 \times 78233 \times 35\,532\,364\,099$$

and that $cr - 1 = 3 \times 18483 \times (p-1)(q-1)$. Dividing by 43 removed the factor of $q - 1$, so t is divisible by $p - 1$ but not $q - 1$, and the attack quickly finds p.

[22] As mentioned in lectures this is a terrible idea: the binary form of n, namely $1\,\overset{75\,\text{zeros}}{\ldots}\,101011\,\overset{72\,\text{zeros}}{\ldots}\,11011101$ makes the factors easily guessable.

Extra: Post-quantum cryptography. Computers operate on the bits $\{0, 1\}$, and binary words made up of these bits. Quantum computers operate instead on *qubits*: a typical qubit is a 'superposition' of 0 and 1. For example, in the standard notation, $|0\rangle + |1\rangle$ is a qubit that, when measured, is equally likely to collapse to each of the classical bits 0 and 1. Two entangled qubits are the quantum analogue of a classical binary word in $\mathbb{F}_2^2$.

In theory, quantum computers can perform some computations far more quickly than classical computers. In particular, using Shor's Algorithm, a quantum computer can quickly find the order of elements in abelian groups. This gives a way to factor n, similar to the attack just seen. It also makes the Discrete Logaithm Problem very easy to solve. However to factor a 2048 bit RSA number n into its two primes p and q requires a quantum computer with at least 2048 qubits. In March 2018, Google reportedly tested a quantum processor with 72 qubits. There is an ongoing debate over whether large scale quantum computing is feasible: it is possible we will find out within our lifetimes.

Because of this NIST is running a competition to choose a public key cryptosystem resistant to quantum attacks. (A similar competition led to the block cipher AES. Symmetric cryptosystems such as AES are widely believed to be resistant to quantum attacks.) Proposals have been submitted using the mathematics of error-correcting codes, lattices and elliptic curves. Much interesting work has been done on evaluating these cryptosystems: it is an exciting time for cryptography.

12. Digital signatures and hash functions

Question. Since anyone can send a message to Alice using her public key, how can Alice be sure she is communicating with Bob?

In this section we suppose the possible messages are elements of $\mathbb{N}_0$. Using the ASCII encoding (see Question 2 on Problem Sheet 5), any English message can be put in this form.

Digital signatures. Suppose Alice and Bob have RSA keys and encryption and decryption functions as shown below. We write d_A and d_B in red, to emphasise that the private key is used in these functions.

	public	private	encrypt	decrypt
Alice	(m, a)	(p, q, r)	$x \xrightarrow{e_A} x^a \bmod m$	$y \xrightarrow{d_A} y^r \bmod m$
Bob	(n, b)	$(?, ?, s)$	$x \xrightarrow{e_B} x^b \bmod n$	$y \xrightarrow{d_B} y^s \bmod n$

Suppose Alice wants to tell Bob her bank details in a message x. She looks up his public key (n, b) and sends him $e_B(x) = x^b \bmod n$. (Assume that $x < n$.)

Eve, the eavesdropper, or Malcolm, the man-in-the-middle, cannot decrypt $x^b \bmod n$, because they do not know s. However Eve can send another message $x'^a \bmod m$, attempting to convince Bob that Alice's first message was wrong. Malcolm has control of the channel and so can replace $x^b \bmod n$ with another $x'^b \bmod n$, again with x' of his choice.

Eve and Malcolm can do this because they know Alice's public key. For comparison, using a symmetric cipher such as DES or AES, only Alice and Bob know the encryption function e_k, so eavesdroppers cannot attack in this way.

How can Bob be confident that a message signed 'Alice' is from Alice, and not from Eve or Malcolm pretending to Alice?

Example 12.1. Bob is expecting a message from Alice. He receives z, and computes $d_B(z) = z^s \bmod n$, but gets garbage. Thinking that Alice has somehow confused the keys, he computes $e_A(z) = z^a \bmod m$, and gets $x \in \mathbb{N}_0$. He then finds that x is the ASCII encoding of

'Dear Bob, my account number is 40081234, best wishes, Alice'.

(a) How did Alice compute z?
(b) Should Bob believe z was sent by Alice?
(c) Can Malcolm read z?
(d) How can Alice avoid the problem in (c)? (Assume that $m < n$.)

Let $x \in \mathbb{N}_0$ be Alice's message. If Alice's RSA modulus m is about 2^{2048} then the message x is a legitimate ciphertext only if $x < 2^{2048}$. This may seem big, but, using the 8-bit ASCII coding, it means only $2048/8 = 2^8 = 256$ characters can be sent. Alice can get round this by splitting the message into blocks, but computing $d_A(x^{(i)})$ for each block $x^{(i)} \in \{1, \ldots, n-1\}$ is slow. It is better to send x, and then append $d_A\big(h(x)\big)$ where $h(x) \in \{0, 1, \ldots, n-1\}$ is a hash of x.

Hash functions and the birthday paradox.

Definition 12.2.

(i) A *hash function* of length r is a function $h : \mathbb{N}_0 \to \mathbb{F}_2^r$. The value $h(x)$ is the *hash* of the message $x \in \mathbb{N}_0$.

(ii) Let (m, a) be Alice's public key in the RSA cryptosystem where $m > 2^r$. To *sign* a message x, Alice computes $h(x) \in \mathbb{F}_2^r$ and, reading $h(x)$ as a number written in binary, computes $d_A\big(h(x)\big)$. The pair $\big(x, d_A(h(x))\big)$ is a *signed message of x from Alice*.

Bob (or anyone else) *verifies* that a pair (x, v) is a valid signed message from Alice by checking that $h(x) = e_A(v)$. Note that x need not be a plaintext: it could be a ciphertext encrypted for Bob.

A cryptographically useful hash function has the following properties:

(a) It is fast to compute $h(x)$.
(b) Given a message $x \in \mathbb{N}_0$, and its hash $h(x)$, it is hard to find $y \in \mathbb{N}_0$ such that $y \neq x$ and $h(y) = h(x)$. (*Preimage resistance.*)
(c) It is hard to find $x, x' \in \mathbb{N}_0$ with $x \neq x'$ such that $h(x) = h(x')$. (*Collision resistance.*)

When Bob receives the signed message (x, v) from Alice, he verifies that $h(x) = e_A(v)$, and so $v = d_A(h(x))$. He now knows that Alice has decrypted (that is signed), the hash value $h(x)$. Only Alice can do this. So an attacker who wants to change x has to replace x with some x' with $h(x') = h(x)$. By preimage resistance, it is hard for the attacker to find any such x'. Therefore Bob can be confident that x really is Alice's message.

A good hash function of length r behaves like a random function from $\mathbb{N}_0$ to $\mathbb{F}_2^r$.

Example 12.3. Malcolm has intercepted a signed message (x, v) from Alice. If he can find x' with $h(x') = v$ then he can replace x with x' and Bob will still verify Alice's signature. Given a hash value v, a brute-force search for x' such that $h(x') = v$ will succeed on each $x' \in \mathbb{N}_0$ with probability $\frac{1}{2^r}$. After hashing 2^r numbers, Malcolm can expect one success.

Thus in (b) 'hard to find' means 'requires at least 2^r hashes'.

Exercise 12.4. Let $h : \mathbb{N}_0 \to \mathbb{F}_2^r$ be a good hash function. On average, how many hashes does an attacker need to calculate to find $x, x \in \mathbb{N}_0$ with $x \neq x'$ and $h(x) = h(x')$?

Thus in (c) 'hard to find' means 'requires at least $2^{r/2}$ hashes'.

The mathematics behind Exercise 12.4 is the well-known Birthday Paradox: in a room with 23 people, the probability is about $\frac{1}{2}$ that two people have the same birthday.

Hash functions in practice. A block cipher with keyspace $\mathbb{F}_2^\ell$ and block size n can be used as a hash function. Fix (and make public, maybe as part of your message) an initialisation state $z^{(0)} \in \mathbb{F}_2^n$. Chop the message x (assumed converted to binary) into binary words $x^{(1)}, x^{(2)}, \ldots, x^{(t)} \in \mathbb{F}_2^\ell$. Then use the block cipher with *keys* $x^{(i)}$, starting with $z^{(0)} \in \mathbb{F}_2^\ell$ as follows:

$$\begin{aligned} z^{(1)} &= z^{(0)} + e_{x^{(1)}}(z^{(0)}) \\ z^{(2)} &= z^{(1)} + e_{x^{(2)}}(z^{(1)}) \\ &\vdots \\ z^{(t)} &= z^{(t-1)} + e_{x^{(t)}}(z^{(t-1)}) \end{aligned}$$

The final state $z^{(t)} \in \mathbb{F}_2^r$ depends on the entire message x in a complicated way, so is a good choice for $h(x)$. Using RSA, Alice sends the signed message $\big(x, d_A\big(h(x)\big)\big)$.

Exercise 12.5. Suppose that we use AES (with 128-bit keys) to hash a message $x \in \mathbb{F}_2^{128}$, using the initialisation state $z^{(0)} = 0 \ldots 0 \in \mathbb{F}_2^{128}$. Then only one step is needed above and the hash value is $h(x) = e_x(0 \ldots 0)$. An adversary therefore knows the plaintext $0 \ldots 0$ and its encryption using x as the key. Why is it hard for her to find x?

Example 12.6. Alice flips a coin and records the result. Bob guesses heads or tails and Alice informs him whether he is correct. If the two can communicate only by email, how can Bob be sure that Alice does not falsely claim that the flip is the opposite of Bob's guess?

Example 12.7 (SHA-256)**.** SHA-256 is the most commonly used hash function today. It has length 256. There is an internal state of 256 bits, divided into 8 words of 32 bits. The message x is chopped into 512 bit blocks; each block is then further divided into words, which are combined by multiplying bits in the same positions (this is 'logical and'), addition in $\mathbb{F}_2^{32}$, cyclic shifts (like an LFSR), and addition modulo 2^{32}, over 64 rounds. As in Cipher Block Chaining, the output for block $x^{(i)}$ is used in the calculation for $x^{(i+1)}$. The best attack is subexhaustive for preimages (b) when the number of rounds is reduced to 57, and subexhaustive for collisions (c) when the number of rounds is reduced to 46.

Extra: hashing passwords. When you create an account online, you typically choose a username, let us say 'Alice' and a password, say 'alicepassword'. A well-run website will not store your password. Instead, oversimplifying slightly, your password is converted to a number x and its SHA-256 hash $h(x)$ is stored. By (b), it is hard for anyone to find another word whose hash is also $h(x)$.

Provided your password is hard to guess, your account is secure, and you have avoided telling the webmaster your password.

Exercise 12.8. As described, it will be obvious to a hacker who has access to the password database when two users have the same password. Moreover, if you use the same password on two different sites, the same hash will be stored on both. How can this be avoided?

Extra: the Bitcoin blockchain.

Example 12.9. The bitcoin blockchain is a distributed record of all transactions involving bitcoins. When Alice transfers a bitcoin b to Bob, she posts a public message x, saying 'I Alice give Bob the bitcoin b', and signs this message[23], by appending $d_A(h(x))$, to get $(x, d_A(h(x)))$.

Signing the message ensures that only Alice can transfer Alice's bitcoins. But as described so far, Alice can double-spend: a few seconds later she can sign another message y, where y says 'I Alice give Charlie the bitcoin b'.

To avoid this, transactions are *validated* in *blocks*. To validate a block of transactions

$$(x^{(1)}, d_{A^{(1)}}(h(x^{(1)}))), (x^{(2)}, d_{A^{(2)}}(h(x^{(2)}))), \ldots$$

a *miner* searches for $c \in \mathbb{N}$ such that, when the list with c appended is converted to a number, its hash, by two iterations of SHA-256, has a large number of initial zeros. (See the following exercise.) Assuming that SHA-256 has property (b), preimage resistance, there is no better way to do this then an exhaustive search for c. The list of validated transactions becomes a *block*; making a new block is called 'growing the blockchain'.

When Charlie receives $(y, d_A(h(y)))$, he looks to see if there is are blocks already containing a transaction involving the bitcoin b mentioned in y. When Bob finds $(x, d_A(h(x)))$ as part of a block with the laboriously computed c, Bob knows Alice has cheated.

Miners are incentivized to grow the block chain: the reward for growing the blockchain is given in bitcoins. Thus bitcoin, which really is nothing more than the blockchain, depends on the computational difficulty of finding preimages and collisions for hash functions. The prize for growing the block chain is only given for blocks that have a consistent transaction history, so Alice's double-spending transaction will not make it into a block.[24]

[23] Rather than use RSA, Bitcoin specifies the ECDSA signature algorithm: very roughly this replaces the ring $\mathbb{Z}_n$ with an elliptic curve. The hash function h is two iterations of SHA-256.

[24] This is a oversimplification: it is possible for two inconsistent blocks to enter the block chain, if they are mined at almost the same time. Then some miners will work on growing the history from block A, and others from block B. The prize for growing the blockchain is only paid for growing the *longest* (consistent) chain. So after a few more verifications the network will agree on one consistent history. In the *Finney attack*, which assumes Alice has considerable computational power, she can (a) mine, but not release, a block verifying a transfer to Charlie (and a number of other, unrelated transactions); (b) make another transaction transferring the same bitcoin to Bob; (c) release her mined block, voiding the transfer to Bob. Bob can avoid being the victim of this attack by waiting for at least one verification of the transfer. It is usual to wait for six.

Miners are further incentivized by transaction fees, again paid in bitcoins, attached to each transaction. These will become more important as the per block reward gets smaller.

An excellent introductory video on bitcoin is available here: `www.youtube.com/watch?v=bBC-nXj3Ng4&feature=youtu.be`. The best summary account of bitcoin is still the original paper: `bitcoin.org/bitcoin.pdf` by Satoshi Nakamoto (2008).

Exercise 12.10. As we saw in Example 12.7, the SHA-256 hash function h takes values in $\mathbb{F}_2^{256}$. Oversimplifying things slightly, to validate a block of bitcoin transactions a miner must find $x \in \mathbb{F}_2^{256}$ such that $h(h(x))$ begins with 72 zeros (in the usual binary form). A modern general purpose microprocessor requires about 128 cycles to calculate a single $h(x)$, and runs at 4GHz, so executes 4×10^9 cycles per second. In 2019, a special purpose device (the ASIC-based 'Antminer R4') advertised on the web for \$1000 claims 8.6×10^{12} hashes per second.

(a) Estimate the time required to validate a block using both systems.
(b) The reward for the miner is 12.5 bitcoins (the reward is halved every 210,000 blocks) plus the transaction fees for all the transactions in the newly validated block. What are the implication for the bitcoin economy?

Extra: main-in-the-middle attack on Diffie–Hellman key exchange. As outlined in §11, Diffie–Hellman key exchange is vulnerable to a man-in-the-middle attack, similar to the attack we saw on RSA. In the scheme in Example 11.1, Malcolm interferes with step (2):

(2′) When Alice sends $2^a \bmod p$ to Bob, Malcolm replaces it with $2^{a'} \bmod p$. When Bob sends $2^b \bmod p$ to Alice, Malcolm replaces it with $2^{b'} \bmod p$.

Now Alice computes $(2^{b'} \bmod p)^a \bmod p$, that is $2^{ab'} \bmod p$, and Bob computes $(2^{a'} \bmod p)^b \bmod p$, that is $2^{a'b} \bmod p$. Malcolm knows $2^a \bmod p$, $2^b \bmod p$ and a' and b', so he can also compute these. When Alice sends a message to Bob using the key derived from $2^{ab'} \bmod p$, Malcolm can decrypt it, read it, and then re-encrypt it using the key derived from $2^{a'b} \bmod p$. He can even change the message if he wishes.

This attack can be avoided using digital signatures. Suppose Bob has RSA encryption and decryption functions e_B and d_B, as in the table at the start of this section. In Step (2), Bob makes a message s by concatenating $2^a \bmod p$ and $2^b \bmod p$, and then sends both $2^b \bmod p$ and $d_B(s)$ to Alice. If Bob has in fact received $2^{a'} \bmod p$, making his message s', Alice will realise this when she verifies the signature and finds that $e_B(d_B(s')) = s' \neq s$. For a refined version of this search for 'Station-to-station' protocol on the web, and see `https://tinyurl.com/yypvycon`.