MT362/462/5462 CIPHER SYSTEMS

MARK WILDON

These notes are intended to give the logical structure of the course; proofs and further examples and remarks will be given in lectures. Further installments will be issued as they are ready. All handouts and problem sheets will be put 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 lectures, and to talk to me after lectures and in my office hours. I am also happy to answer questions about the lectures or problem sheets by email. My email address is `mark.wildon@rhul.ac.uk`.

Lectures: Monday 4pm (MFLEC), Friday 11am (MC219), Friday 4pm (MC219).

Extra lecture for MSc students doing MT5462: Thursday 1pm (MC336).

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

Date: First Term 2018/19.

CIPHER SYSTEMS

We will study symmetric and public key ciphers, understand how they promise confidential communication, and 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 Kerckhoff's Principle.

(B) *Stream ciphers:* linear feedback shift registers and pseudo-random number generation.

(C) *Block ciphers:* design principles, Feistel networks, DES and AES.

(D) *Public key ciphers and digital signatures:* one-way functions, Diffie–Hellman, RSA and ElGamal. Factoring and discrete logs. Hash functions and certificates. 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 MSc topics.

Also you will find a link on Moodle to Dr. Siaw-Lynn Ng's notes. These will give you a different view of the course material. Highly recommended.

Problem sheets. There will be 8 marked problem sheets; the first is due in on Monday 15th October. Answers to the preliminary problem sheet will be posted on Moodle on Monday 8th October.

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 `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 has access!**

Exercises in these notes. Exercises set in these notes are mostly simple tests that you are following the material. Some will be used for quizzes in lectures. **Doing the others will help you to review your notes.**

Optional questions and extras. Any 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, know the definitions and main results from lectures, and can prove the results whose proofs are marked as examinable in these notes, then you should do very well in the examination.

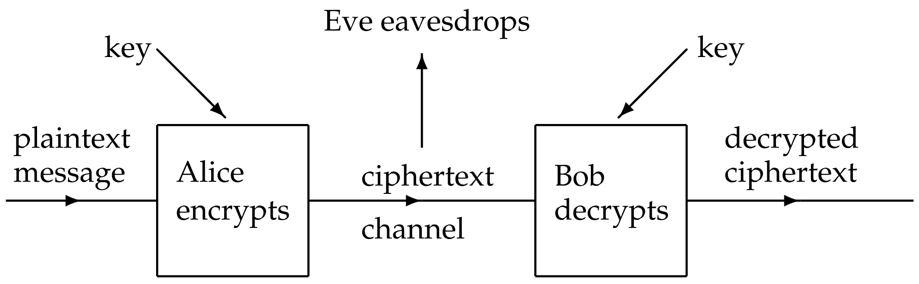
(A) Introduction: alphabetic ciphers and the language of cryptography

1. Introduction: security and Kerckhoff's Principle

Lecture 1

This course is about the mathematics underlying cryptography. But it is only sensible to have some idea of the overall goal!

As a basic model, Alice wants to send Bob a *plaintext* message. This message may be intercepted in the channel by the eavesdropper Eve, or even 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*.



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) By default email is unencrypted. Each email will typically be received and sent on by multiple computers on the internet 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 care most about 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, the bank's main security requirements are authentication and non-repudiation. It is good practice to use two-factor authentication, so 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.

Kerckhoff's Principle. It is obviously important in cryptography to be very clear about what is public information and what is private. Kerckhoff'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. (We will make this more precise later.)

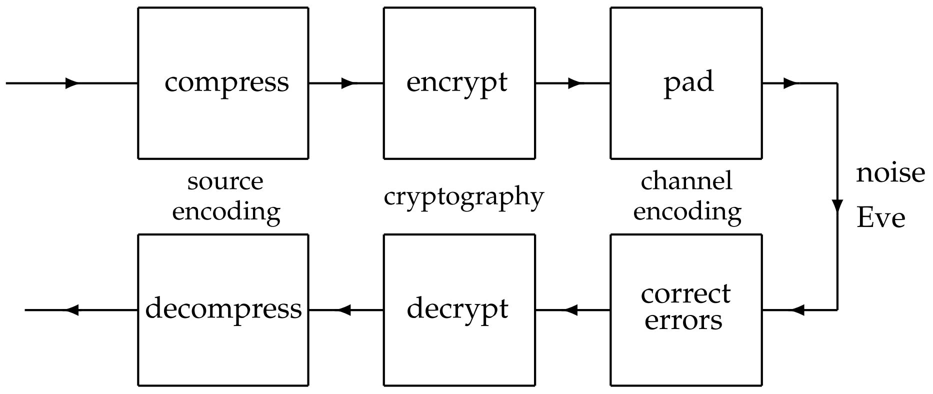
Example 1.2. On Friday, Alice will learn Bob's final year exam result. This is a plaintext $x \in \{0, 1, \ldots, 99\}$. The two trust their friend Trevor.

- On Monday, Trevor chooses a key $k \in \{0, 1, \ldots, 99\}$ and gives it to Alice and Bob in person.
- On Tuesday, Bob leaves for his planned trip to Botswana. He can read email. Bob cannot send email or communicate in any other way.
- On Friday, Alice will email Bob the ciphertext $(x + k)$ mod 100.

By Kerckhoff's Principle, all this, except for the value of k, is known to the whole world.

(a) Can Eve, the eavesdropper, learn anything from Alice's email?
(b) Despite the good news in (a), the scheme has its flaws. Find some problems with it.
(c) 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 MT361/461/5461 Error Correcting Codes. But there is no need to do these courses to understand this one.



2. Alphabetic ciphers

We begin with some ciphers that operate directly on English letters and words. It is a useful convention in this section to write plaintexts in lower case and ciphertexts in upper case.

Caesar and substitution ciphers.

Example 2.1. The *Caesar cipher* with key $s \in \{0, 1, \ldots, 25\}$ encrypts a word (of any length) 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 is always broken by knowledge of a plaintext/ciphertext pair.

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

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

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

Exercise 2.4. How many substitution ciphers are there?

Lecture 2

A sufficient long ciphertext can be decrypted by using frequency analysis to deduce $\pi(\mathrm{e}), \pi(\mathrm{t}), \ldots$, and then guessing likely words. Even the 10 character message above has 'e' as its most common character. Some common digraphs and trigraphs are 'th', 'he', 'in', 'er', 'the', 'ing', 'and'.

The table below (taken from Stinson's book) 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

Example 2.5. Eve intercepts 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 relative 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 `AlphabeticCiphers` 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 `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.

Lecture 3

Exercise 2.6.

(a) After deciphering in Example 2.5, Eve knows that $\pi(a) = \texttt{U}$, $\pi(b) = \texttt{T}$, 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) Suppose Mark can encrypt a plaintext of his choice using e_π. This is a 'chosen plaintext account': see §3. What is the simplest way for him to learn π?

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

Example 2.7. To make this process automatic, we need a quantitative way to measure how 'close to English' $e_\tau^{-1}(y)$ is. A good scoring function is $\sum_q \log p_q$ where the sum is over all quadgrams in $e_\tau^{-1}(y)$ and p_q is the probability of the quadgram q. (This is motivated by maximum likelihood estimation.) For example, the most common quadgram is 'tion', with $p_{\text{tion}} = 0.00312$, followed by 'nthe' and 'ther'. Note the score is always negative, since $\log p_q < 0$ for each quadgram q.

As usual 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 swapping two chosen cards, this means all 26! (see Exercise 2.4) possible keys can be reached, by taking enough steps. For instance in Example 2.5 we start with the guess

$$\tau(\mathrm{a}) = \texttt{U}, \tau(\mathrm{b}) = \texttt{V}, \tau(\mathrm{c}) = \texttt{F}, \tau(\mathrm{d}) = \texttt{L}, \tau(\mathrm{e}) = \texttt{X}, \ldots,$$
$$\ldots, \tau(\mathrm{p}) = \texttt{B}, \ldots, \tau(\mathrm{y}) = \texttt{D}, \tau(\mathrm{z}) = \texttt{A}$$

implying the plaintext is

ile gutmafetiah owkenirve og nspciobsaclp rd io etawhe iyo ceoche uduahhp segessem io ad ahrne atm wow io noffutr-naie oves at rtdenuse nlatteh rt dunl a pay ilai at occoteti . . .

The first step is chosen to maximize the increase in the score. It turns out to be optimal to swap the encryptions of 'p' and 'y'. The new guess for the key is τ' where $\tau'(p) = \texttt{D}$ and $\tau(y) = \texttt{B}$; otherwise τ' agrees with τ. (**Minor correction:** I forgot the convention that ciphertext letters are written in upper case in the guess for τ on the previous page.)

After 16 steps the implied plaintext is

> rle gutfametrah owvecrike og csyprobsaply id ro etawhe rno peophe uduahhy segessef ro ad ahice atf wow ro commuticare okes at itdecuse clatteh it ducl a nay rlar at oppotetr ...

and after 31 steps the implied plaintext is correct in every character! Try it online at `repl.it/@mwildon/SubstitutionHillClimbWeb`.

Exercise 2.8. The strategy in Example 2.7 is called 'hill-climbing'. Why this name? (This, and the associated weakness of the substitution cipher, is all you need to remember from Example 2.7.)

Another reason why the substitution cipher is weak is because 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

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

Using this bijection we identify a word of length ℓ with an element of $\{0, 1, \ldots, 25\}^\ell$. For example, '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 word. Suppose that k has length ℓ. Given a plaintext x with its spaces deleted, we define its encryption by

$$e_k(x) = (x_1 + k_1, x_2 + k_2, \ldots, x_\ell + k_\ell, x_{\ell+1} + k_1, \ldots)$$

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

Example 2.10. Take $k = \mathrm{emu}$, so k has length 3. Under the bijection between letters and numbers, emu $\longleftrightarrow (4, 12, 20)$. The table overleaf shows that

$$e_{\mathrm{emu}}(\texttt{meetatmidnightnear}) = \texttt{QQYXMNQUXRUALFHIML}.$$

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
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	e	m	u	e	m	u	e	m	u	e	m	u	e	m	u	e	m	u
	4	12	20	4	12	20	4	12	20	4	12	20	4	12	20	4	12	20
$x_i + k_i$	16	16	24	23	12	13	16	20	23	17	20	0	11	5	7	8	12	11
	Q	Q	Y	X	M	N	Q	U	X	R	U	A	L	F	H	I	M	L

Lecture 4

Exercise 2.11.

(a) If you had to guess, which of the following would you say was more likely to be the ciphertext from a substitution cipher?

```
KDDLVFUDLNELUHLYJAWLWGLWUJDULF
KYBDRDDFCLVCVEDFLDUVYDKKLZCNPO
KYEYAXBICDMBRFXDLCDPKFXLCILLMO
```

These come from taking every 9th, every 3rd and every position in a ciphertext in Example 2.16 below; it is encrypted using a Vigenère cipher with key length 9.

(b) Why should we expect the split ciphertext from a Vigenère cipher to have the most 'spiky' frequency distribution at the length of the keyword?

This gives some motivation for the following statistic.

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 $I(\texttt{QMUUFM}) = \frac{2}{15}$. What is $I(\texttt{AAABBC})$?

There is a simple formula for $I(y)$. (An examinable proof.)

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*

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

We now have a strategy for decrypting a Vigenère ciphertext.

Attack 2.15. *Given a Vigenère ciphertext, split it into groups by taking every ℓ-th letter for all small ℓ, as in Exercise 2.11. If the ciphertext is long enough, the Index of Coincidence will be greatest at the key length. Each ciphertext split at the key length is the output of a Caesar cipher; assuming the most common letter is the encryption of 'e' determines the shift.*

Example 2.16. The final 554 words (or 2534 characters) of Chapter 1 of *Persuasion* by Jane Austen begin

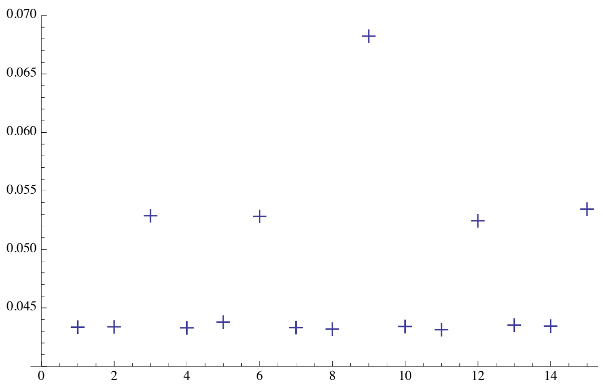
> Such were Elizabeth Elliot's sentiments and sensations; such the cares to alloy, the agitations to vary, the sameness and the elegance, the prosperity and the nothingness of her scene of life; such the feelings to give interest to a long, uneventful residence in one country circle, to fill the vacancies which there were no habits of utility abroad, ...

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

`KYEYAXBICDMBRFXDLCDPKFXLCILLMOVRMCEL` ....

The graph below shows the mean Index of Coincidence when the ciphertext is split by taking every ℓ-th position, for $\ell \in \{1, 2, \ldots, 15\}$. We correctly guess that the length of the key is 9. Taking every 9-th letter of the ciphertext we get '`KDDLVFUDLNELUHLYJA` ...'. The frequency table (as in Example 2.5) begins

W	L	S	K
11.0	10.6	7.4	7.1



Assuming $'W' \longleftrightarrow 22$ is the encryption of $'e' \longleftrightarrow 4$, the shift in the Caesar cipher is $18 \longleftrightarrow 's'$, so we guess the first letter of the key is 's'. The MATHEMATICA notebook on Moodle shows this simple strategy works in all 9 key positions to reveal the key.

Exercise 2.17. Explain why there are smaller peaks at 3, 6, 12 and 15 in the plot of Indices of Coincidence above.

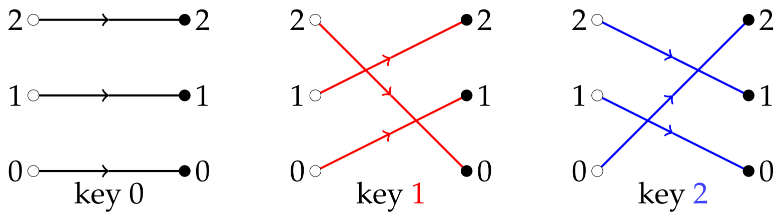
Statistical methods. The f_i^2 in the numerator of the formula in Lemma 2.14 may remind you slightly of the χ^2-test: the connection is explored in the optional Question 7 on Sheet 1.

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.

3. Cryptosystems, attack models and perfect secrecy

Lecture 5

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



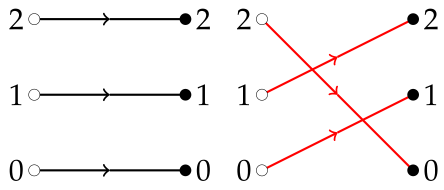
Definition 3.1. Let $\mathcal{K}, \mathcal{P}, \mathcal{C}$ be finite sets. A *cryptosystem* is a family of *encryption functions* $e_k : \mathcal{P} \to \mathcal{C}$ and *decryption functions* $d_k : \mathcal{C} \to \mathcal{P}$, one for each $k \in \mathcal{K}$, such that for each $k \in K$,

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

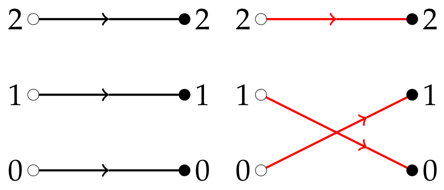
for all $x \in \mathcal{P}$. We call $\mathcal{K}$ the *keyspace*, $\mathcal{P}$ the set of *plaintexts*, and $\mathcal{C}$ the set of *ciphertexts*.

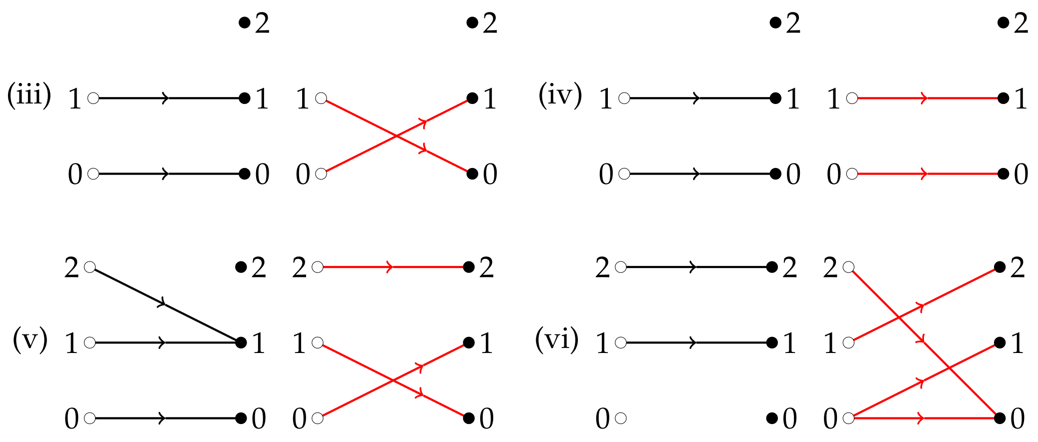
Exercise 3.2. Each diagram (i)–(vi) below each show two functions. Which illustrate the encryption functions in a cryptosystem with two keys (one **black**, one **red**)? In each case $\mathcal{P}$ is on the left-hand side and $\mathcal{C} = \{0, 1, 2\}$ is on the right-hand side.

(i)



(ii)





Please make sure you understand what is wrong with (v) and the two things wrong with (vi).

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 3.3.

(i) Show that e_k is injective for each $k \in \mathcal{K}$.
(ii) Show that if $|\mathcal{P}| = |\mathcal{C}|$ then the encryption functions are bijections and $d_k = e_k^{-1}$ for each $k \in \mathcal{K}$.

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 $7 \times 8 \equiv 1 \bmod 11$. We say that 8 is the *inverse* of 7, modulo 11.

Example 3.4 (Affine cipher)**.** Let p be prime. The *affine cipher* on $\mathbb{Z}_p$ has $\mathcal{P} = \mathcal{C} = \mathbb{Z}_p$ and

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

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

For example, in the affine cipher on $\mathbb{Z}_{11}$, $e_{(7,2)}(5) = 4$ since $7 \times 5 + 2 \equiv 4 \bmod 11$ and, as expected, $d_{(7,2)}(4) = 5$ since $8 \times (4 - 2) \equiv 5 \bmod 11$.

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

It is no real restriction that the plaintext in the affine cipher have to be numbers in $\{0, 1, \ldots, p-1\}$: there are arbitrarily large primes, and it is easy to convert an English plaintext into a number. (The ASCII encoding will be seen on a later problem sheet.)

Lecture 6

Exercise 3.5. Consider the affine cipher on $\mathbb{Z}_5$.

(i) Suppose that Eve observes the ciphertext 2. Does she learn anything about the plaintext?

(iii) Suppose that Mark knows that $e_{(a,c)}(1) = 2$. What does he learn about the key? What happens if he later learns $e_{(a,c)}(0)$?

Attack models. In each of the *attack models* below, we suppose that Alice is sending ciphertexts to Bob encrypted using the key $k \in \mathcal{K}$. The aim of the adversary (Eve or Mark) is to determine all or part of 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 $y = e_k(x)$.
- *Chosen ciphertext.* Mark may choose any $y \in \mathcal{C}$ and is given the decryption $x = d_k(y)$.

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

Remark 3.6.

(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 3.5, the affine cipher requires two choices of plaintext, and by Question 4 on Sheet 1, the substitution cipher and the Vigenère cipher just one.

(3) Later in the course we will see modern 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.*

Probability model. We agreed in Exercise 3.5 that the affine cipher seemed secure against a (single) known ciphertext attack. One way to make this intuitive idea mathematically precise uses probabilities. There are notes on Moodle reviewing basic probability theory.

Fix a cryptosystem in our usual notation. To define a probability space on $\mathcal{K} \times \mathcal{P} \times \mathcal{C}$ we assume 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 message is $x \in \mathcal{P}$ and r_k is the probability the key is k then the probability measure is defined by

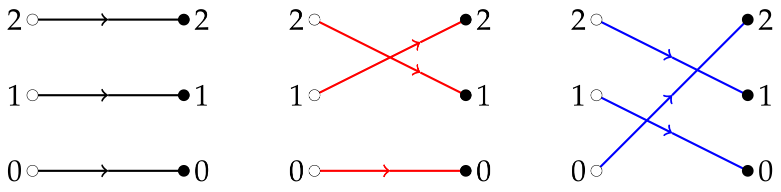
$$p_{(k,x,y)} = \begin{cases} r_k p_x & \text{if } y = e_k(x) \\ 0 & \text{otherwise.} \end{cases}.$$

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

Exercise 3.7. Is the assumption that the key and plaintext are independent reasonable?

Example 3.8.

(a) The cryptosystem below uses three keys from the affine cipher on $\mathbb{Z}_3$. We will use it for a quiz in lectures.



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

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

Note that $\mathbf{P}[Y = y | X = x] = \sum_{k \in \mathcal{K} : e_k(x) = y} \mathbf{P}[K = k]$ is a sum of key probabilities.

(b) In the Caesar cipher on $\{0, 1, 2\}$, shown before Definition 3.1, there are three keys. Suppose keys are chosen with equal probability $\frac{1}{3}$ and, as usual, the probability distribution on plaintexts is p_0, p_1, p_2. We will check that $\mathbf{P}[X = x | Y = y] = p_x$ for all x, $y \in \{0, 1, 2\}$. Knowing the ciphertext tells Eve nothing new about the plaintext. **Lecture 7**

(c) In Exercise 3.2(i), $\mathcal{P} = \mathcal{C} = \{0, 1, 2\}$. Suppose the two keys are used with equal probability $\frac{1}{2}$. We have $\mathbf{P}[Y = 1] = \frac{p_0 + p_1}{2}$ and

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

[1]To be very formal, X, Y and K 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 $Y(k, x, y) = y$.

These probabilities are usually not the same as p_0, p_1, p_2. (Just take $p_2 \neq 0$.) An Eve eavesdropping on ciphertext 1 learns the plaintext is either 0 or 1 (and not 2), and has an idea of how probable each plaintext is.[2]

To summarise: (b) is safe against a known ciphertext attack but (a) and (c) are broken by a known cipher ciphertext attack. **Follow up:** show how to break (b) by a (single) known plaintext attack.

Shannon's Perfect Secrecy Theorem.

Definition 3.9.

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

By Example 3.8(b), the Caesar cipher on $\{0, 1, 2\}$ has perfect secrecy when keys are used with equal probability. If instead $\mathbf{P}[K = 0] = \mathbf{P}[K = 1] = \frac{1}{2}$ and $\mathbf{P}[K = 2] = 0$ we can ignore the blue key 2 and get the cryptosystem in Exercise 3.2(i), which we saw in Example 3.8(c) does not have perfect secrecy.

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

Theorem 3.10 (Shannon 1949)**.** *Suppose a cryptosystem (in our usual notation) has perfect secrecy,* **that** $\mathbf{P}[K = k] > 0$ **for each** $k \in \mathcal{K}$ **[correction!]**, *and that for all* $y \in \mathcal{C}$ *there exists* $x \in \mathcal{P}$ *and* $k \in \mathcal{K}$ *such that* $e_k(x) = y$.

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

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

(c) *Suppose* $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$. *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.*

Some good questions to ask about a theorem are 'What examples of it have I seen?', 'Can the hypotheses be weakened?', 'Does the converse hold?'. These are explored on Problem Sheet 2. In particular, Question 6 asks you to show a converse result: if (c) holds then the cryptosystem has perfect secrecy.

[2] In the language of Bayesian statistics, Eve's posterior probabilities are different to her prior probabilities.

Proof of Theorem 3.10. Let $p_x = 1/|\mathcal{P}|$, so $p_x > 0$ for each $x \in \mathcal{P}$. For $x \in \mathcal{P}$ and $y \in \mathcal{C}$, let

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

For each $y \in \mathcal{C}$ there exists $x^\star \in \mathcal{P}$ and $k^\star \in \mathcal{K}$ such that $k^\star \in \mathcal{K}_{x^\star y}$. Hence $\mathbf{P}[Y = y] \geq \mathbf{P}[X = x^\star, K = k^\star] = \mathbf{P}[X = x^\star]\mathbf{P}[K = k^\star] > 0$ for all $y \in \mathcal{C}$.

Since $\mathbf{P}[Y = y] > 0$ for each $y \in \mathcal{C}$, the conditional probabilities $\mathbf{P}[X = x|Y = y]$ are defined. By perfect secrecy followed by the usual calculation,

$$\begin{aligned} p_x = \mathbf{P}[X = x|Y = y] &= \frac{\mathbf{P}[Y = y|X = x]\mathbf{P}[X = x]}{\mathbf{P}[Y = y]} \\ &= \frac{\mathbf{P}[Y = y|X = x]}{\mathbf{P}[Y = y]} p_x = \frac{\mathbf{P}[k \in \mathcal{K}_{xy}]}{\mathbf{P}[Y = y]} p_x \end{aligned}$$

for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$. Since $p_x > 0$ we deduce

$$(\dagger) \qquad \mathbf{P}[k \in \mathcal{K}_{xy}] = \mathbf{P}[Y = y] > 0.$$

Therefore $\mathcal{K}_{xy}$ is non-empty for each $x \in \mathcal{P}$ and $y \in \mathcal{C}$, as required for (a).

Let $x \in \mathcal{P}$. Observe that $\mathcal{K} = \bigcup_{y \in \mathcal{C}} \mathcal{K}_{xy}$ where the union is disjoint. By (a) each $\mathcal{K}_{xy}$ is non-empty. Therefore

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

giving (b).

In (c) we have $|\mathcal{K}| = |\mathcal{C}|$. Hence each inequality in ($\ddagger$) is an equality, and so $|\mathcal{K}_{xy}| = 1$ 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 $\mathcal{K}_{xy}$) such that $e_k(x) = y$. Fix $y^\star \in \mathcal{C}$ and, for each $x \in \mathcal{P}$, let k_x be the unique key such that $e_{k_x}(x) = y^\star$. If $k_x = k_{x'} = k$ then **Lecture 8**

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

But e_k is injective, hence $x = x'$. Therefore the keys k_x for $x \in \mathcal{P}$ are distinct, and since $|\mathcal{P}| = |\mathcal{K}|$, these are all the keys. By ($\dagger$), $\mathbf{P}[k = k_x] = \mathbf{P}[k \in \mathcal{K}_{xy^\star}] = \mathbf{P}[Y = y^\star]$ is independent of $x \in \mathcal{P}$. Therefore each key is used with equal probability. $\square$

Latin squares. Consider a cryptosystem with perfect secrecy in which $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0, 1, \ldots, n-1\}$. **[Correction: $\mathcal{C}$ and $\mathcal{K}$ not $|\mathcal{C}|$ and $|\mathcal{K}|$.]** By (c) in Theorem 3.10, for all 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

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

The Caesar cipher on $\{0, 1, 2\}$ seen at the start of this section has matrix

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

Note rows and columns are numbered from 0 rather than the usual 1. Conversely, 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.

4. Entropy and key uncertainty

Lecture 9

Motivation for Entropy. Suppose Bob picks $x \in \{0, 1, \ldots, 15\}$. How many yes/no questions does Alice need to guess x? Question 2 on the Preliminary Problem Sheet gives 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 4.1. Explain why no questioning strategy can guarantee to use fewer than four questions.

Example 4.2. 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 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) $p_0 = \frac{1}{2}$, $p_1 = \frac{1}{4}$, $p_2 = \frac{1}{8}$, $p_3 = \frac{1}{8}$.

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

Definition 4.3. Let $\mathcal{X}$ be a finite set.

(i) The *entropy* of a probability distribution p_x on $\mathcal{X}$ is

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

[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.

(ii) 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 $\lim_{p\to 0} -p \log_2 p = 0$.

Exercise 4.4.

(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 4.2(b) then

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

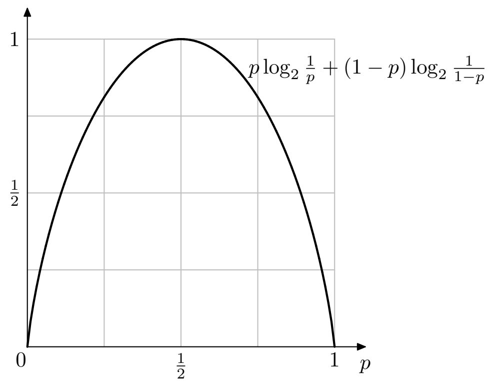
Show that in all three cases, $H(p)$ is the average number of questions, using the strategy found in this exercise.[4]

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 4.2(a) is 2 bits.

Example 4.5.

(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 below.



[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 Channels.

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 3.2(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 $\mathbf{P}[K = \text{black}] = 1 - r$. 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 hence find $H(Y)$ when $r = 0, \frac{1}{4}, \frac{1}{2}$. Is it surprising that usually $H(Y) > H(X)$?

Lecture 10 *Conditional entropy and key uncertainty.*

Definition 4.6. 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 $\mathcal{K}\times\mathcal{C}$. It may also be helpful to note that $H(K|Y)$ is the expected value of $H(K|Y = y)$, as y varies over $\mathcal{C}$.

Example 4.7. 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.

The most important property of conditional entropy is stated in the lemma below. 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 4.8 (Chaining Rule)**.** *Let K and Y be random variables. Then*

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

Lecture 11 We need two further results to prove the main theorem of this section.

Lemma 4.9. *Let K and X be random variables. If K and X are independent then $H(K, X) = H(K) + H(X)$.*

For a proof see Question 1 on Sheet 3.

Lemma 4.10. *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 $H\big(f(Z)\big) = H(Z)$.*

A proof is given below. You may prefer this intuitive version: since f is injective, the non-zero probabilities in the probability distribution of $f(Z)$ are the same as for Z. Since the entropy of a random variable depends only on its probability distribution, the entropies of Z and $f(Z)$ are the same.

Proof. Since f is injective, $\mathbf{P}[Z = z] = \mathbf{P}[f(Z) = f(z)]$ for each $z \in \mathcal{Z}$. If $w \in \mathcal{W}$ and $w \neq f(z)$ for any $z \in \mathcal{Z}$ then $\mathbf{P}[f(Z) = w] = 0$. Therefore the contribution of w to $H(f(Z))$ is $-0 \log_2 0 = 0$. Hence

$$\begin{aligned} H(f(Z)) &= -\sum_{w \in \mathcal{W}} \mathbf{P}[f(Z) = w] \log_2 \mathbf{P}[f(Z) = w] \\ &= -\sum_{z \in \mathcal{Z}} \mathbf{P}[f(Z) = f(z)] \log_2 \mathbf{P}[f(Z) = f(z)] \\ &= -\sum_{z \in \mathcal{Z}} \mathbf{P}[Z = z] \log_2 \mathbf{P}[Z = z] \\ &= H(Z). \end{aligned}$$

as required. □

Theorem 4.11 (Shannon, 1949). *Take a cryptosystem in our usual notation. Then*

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

We end with two applications of Shannon's Theorem.

English entropy and the one-time pad. Let $\mathcal{A} = \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ be the alphabet. We take $\mathcal{P} = \mathcal{C} = \mathcal{A}^n$: you can think of this as the set of all strings of length 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 '`abcdefgh`', '`goodwork`' $\in \mathcal{P}$ but $\mathbf{P}[X_8 = \text{'}\texttt{abcdefgh}\text{'}] = 0$ whereas $\mathbf{P}[X_8 = \text{'}\texttt{goodwork}\text{'}] > 0$.

Shannon estimated that the per-character redundancy of English plaintexts, with spaces, is about 3.200. (See the optional extras for this part.) We shall suppose his estimate is also good for plaintexts in $\mathcal{A}^n$.

Let $R = 3.200$. If English plaintexts of length n had no redundancy, their per-character entropy would be $\log_2 26 \approx 4.700$. Therefore the per-character entropy of English is about $\log_2 26 - R \approx 1.500$, and

$$H(X_n) \approx (\log_2 26 - R)n \approx 1.500n.$$

Lecture 12

Example 4.12 (One-time pad)**.** Fix $n \in \mathbb{N}$. The *one-time pad* is a cryptosystem with plaintexts, ciphertexts and keyspace $\mathcal{A}^n$. The encryption functions are defined by

$$e_k(x) = (x_1 + k_1, x_2 + k_2, \ldots, x_n + k_n)$$

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. Thus the one-time pad is the Vigenère cipher when the key has the same length as the plaintext. For example, when $n = 8$,

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

as shown in the table below

i	1	2	3	4	5	6	7	8
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

Suppose that all keys in $\mathcal{A}^n$ are equally likely. Then all ciphertexts are equally likely, and by Example 4.5(2)

$$H(K) = (\log_2 26)n$$
$$H(Y_n) = (\log_2 26)n.$$

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

$$H(K|Y_n) = H(K) + H(X_n) - H(Y_n) = (\log_2 26 - R)n = H(X_n).$$

Thus when Eve observes the ciphertext Y_n, she is as uncertain about the key as she is about the plaintext.

It is intended that the one-time-pad is used only once. If the same key is used for multiple encryptions, Eve can learn much more.

Example 4.13. The spy-master Alice and her agent Bob have agreed to use the one-time pad, with a randomly chosen key, for emergency messages. Following Kerckhoff's assumption, 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 calculates $\texttt{lxcghyrrroznfy} - \texttt{atcldqezyomuua} = \texttt{leaveinstantly}$. So far Eve has learned nothing, except that Alice has sent Bob the ciphertext $y = \texttt{lxcghyrrroznfy}$. Eve cannot guess Alice's message x: 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. Clearly for every guessed plaintext, there is a possible key. 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 to obtain $\Delta = \texttt{fqsiyenaahwtdl}$. Note that Δ does not depend on k.

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 message). This property is so rare that Eve and her computer[5] can fairly easily deduce x' and $\Delta + x'$, and, from either of these, the key k.

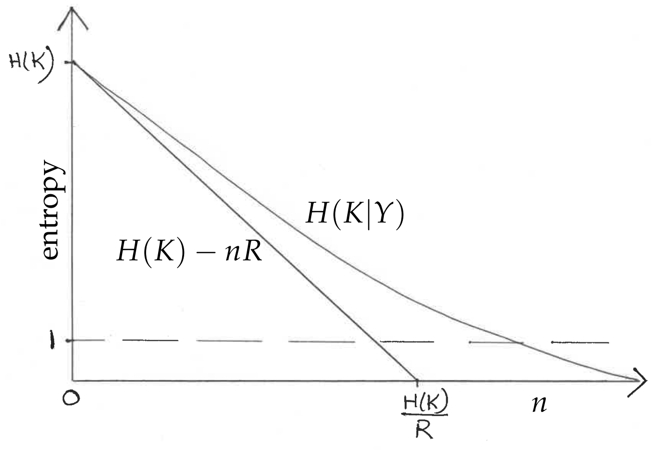
Unicity distance. In Example 4.13 we proved that for the one-time-pad $H(K|Y_n) = (\log_2 26 - R)n$ and that $H(K) = (\log_2 26)n$. Therefore

$$(\star\star) \qquad H(K|Y_n) = H(K) - Rn.$$

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 $\mathcal{P} = \mathcal{C} = \mathcal{A}^n$ and the messages are English texts.

Exercise 4.14. What is the largest length of ciphertext n for which $(\star\star)$ could hold with equality?

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



[5]The code used in lectures is here: `https://repl.it/@mwildon/OneTimePad2`.

Definition 4.15. 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.

Exercise 4.16. In the substitution cipher attack in Example 2.5 we saw that the ciphertext y of length 280 determined the key π except for $\pi(\texttt{k})$, $\pi(\texttt{q})$, $\pi(\texttt{z})$. We saw in Exercise 2.6(a) that $\pi(\texttt{k})$, $\pi(\texttt{q})$, $\pi(\texttt{z})$ are the three letters, namely `A`, `E`, `N`, which never appear in the ciphertext. Assuming equally likely keys, what is $H(K|Y_{280} = y)$? What is $H(K)$?

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

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

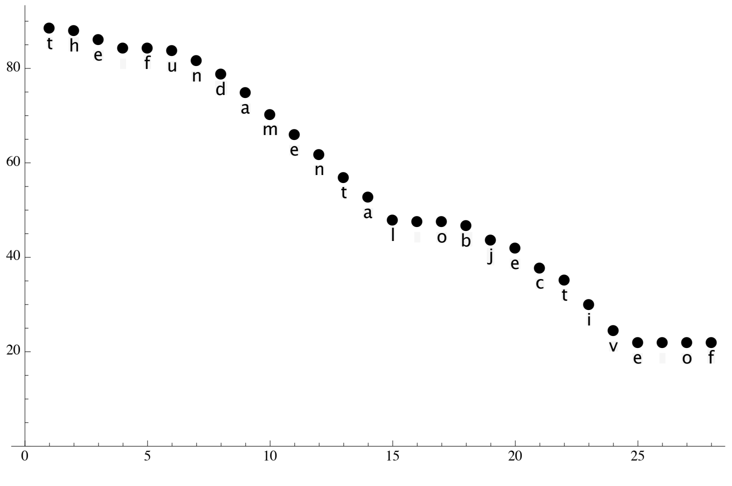
So 28 characters of ciphertext should, in theory, determine most of the key.

Example 4.17. 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 gives 6 possible decryptions of the first 24 letters. These include '`imo purgatorial hedonics`', '`iwo purgatorial hedonism`' and '`the fundamental objectiv`'. 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, the argument in Exercise 4.16 shows that $H(K|Y = y_{28}) = \log_2 10! = 21.791$. Nothing new about the key is learned after letter 25, so this is the value of the final 4 points in the graph of $H(K|Y_n)$ for $1 \leq n \leq 28$ below.



Extra: one idea in Shannon's argument for unicity distance. The following exercise appears on the Preliminary Problem Sheet.

Exercise 4.18. 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 4.19 below.

Extra: Shannon's argument for unicity distance. Shannon's Noisy Coding Theorem states that given a binary channel of capacity C, one can communicate at any chosen rate $R < C$ with an asymptotically negligible probability of error. Shannon's amazing insight was that a good way to do this is to choose a binary code of rate R *at random*.

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, and a keyspace $\mathcal{K}$. The encryption functions are random permutations. Thus for each $k \in \mathcal{K}$, we have a random bijection $e_k : \mathcal{P} \to \mathcal{P}$.

As a simple model for English plaintexts, suppose that $\mathcal{P} = \mathcal{P}_{\text{common}} \cup \mathcal{P}_{\text{rare}}$ where the union is disjoint. Let $|\mathcal{P}_{\text{common}}|/|\mathcal{P}| = c$. We suppose that each common plaintext is sent with equal probability $1/|\mathcal{P}_{\text{common}}|$, so rare plaintexts are never sent.

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

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

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 $g(y) \geq 1$.

Lemma 4.19. $g(y) \sim 1 + \mathrm{Bin}(c, |\mathcal{K}| - 1)$.

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 $\mathrm{Bin}(c, |\mathcal{K}| - 1)$. Now add 1 to count $k^\star$. $\square$

Since the keys are equiprobable, when Eve observes $y \in \mathcal{C}$, her uncertainty in the key is $\log_2 g(y)$. That is, $H(K|Y = y) = \log_2 g(y)$. By the formula for conditional expectation,

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

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 $Z \sim \mathrm{Bin}(c, |\mathcal{K}|)$. 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: Shannon's game. Shannon's estimate of 1.5 bits for the per-character entropy of English comes from his paper, *Prediction and entropy of printed English*, Bell System Technical Journal, **30** (1951) 50–64. In it he invented, and got human subjects to play, the following ingenious game.

Imagine a computer is programmed to guess English plaintexts. It is told as soon as it guesses a character correctly, and then moves on to the next. For example if the plaintext is '`information`', the computer might guess

- `e, t, a, i` (wrong, wrong, wrong, correct: plaintext starts `i`),
- `t, s, n` (wrong, wrong, correct: plaintext starts `in`),
- `t, f` (wrong, correct: plaintext starts `inf`)

and then get every subsequent character right immediately, except for a misguess of ␣ (space) rather than `a` on character 7. The algorithm the computer uses is unknown, but could be very complicated, perhaps using a huge library of stored texts. It is deterministic: given the same plaintext, the computer always makes the same guess.

Exercise 4.20.

(a) Explain why given the sequence $(4,3,2,1,1,1,2,1,1,1,1)$ and access to the computer, you can reconstruct the plaintext.

(b) Suppose you sample a random variable G and get the sequence of values in (a). Estimate the entropy of G. Why is $H(G)$ an upper bound on the entropy of the plaintext?

See the answer to Question 6 on Sheet 3 for a solution. You can play the game online at `repl.it/@mwildon/ShannonGuess2py`.

(B) Stream ciphers

5. LINEAR FEEDBACK SHIFT REGISTERS

Computers are deterministic: given the same inputs, you always get the same answer. In this part we will see how to get sequences that 'look random' out of deterministic algorithms. We will use these sequences to define cryptosystems, and see how they may be attacked. We will also see some ways to define randomness more precisely.

Lecture 13

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 seen here, 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.

Definition of LFSRs.

Definition 5.1.

(i) Let $\ell \in \mathbb{N}$. A *linear feedback shift register* of *width* ℓ with *taps* $T \subseteq \{0, 1, \ldots, \ell-1\}$ is a function $F : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell$ of the form

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

(ii) The function $f : \mathbb{F}_2^\ell \to \mathbb{F}_2$ defined by $f(x) = \sum_{t \in T} x_t$ 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

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

Equivalently, $k_s = \sum_{t \in T} k_{s-\ell+t}$. Thus an LFSR shifts the bits in positions 1 to $\ell - 1$ left, and puts a new bit, defined by its feedback function, into the rightmost position $\ell - 1$. Taking all these rightmost positions gives the keystream. **This very useful property is expressed by**

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

Here F^s is the function defined by applying F a total of s times.

Example 5.2. The LFSR F of width 4 with taps $\{0, 1\}$ 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 $k = 0111$ **[Typo of** 01111 **corrected]** is

$$\begin{array}{l}(0,1,1,1,1,0,0,0,1,0,0,1,1,0,1,0,1,1,1,1\ldots)\\ \;\;0\;\;1\;\;2\;\;3\;\;4\;\;5\;\;6\;\;7\;\;8\;\;9\;\;0\;\;1\;\;2\;\;3\;\;4\;\;5\;\;6\;\;7\;\;8\;\;9\end{array}$$

repeating from position 15 onwards: $k_s = k_{s+15}$ for all $s \in \mathbb{N}_0$.

(iii) *Exercise:* observe that $k' = 0001$ appears in positions 5, 6, 7, 8 of the keystream above. Find the keystream for k'.

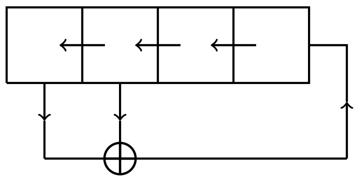
(iv) Starting with $k = 0111$, the sequence k, $F(k)$, $F^2(k)$, $F^3(k), \ldots,$ $F^{14}(k)$, $F^{15}(k)$ is

$$0111, 1111, 1110, 1100, 1000, 0001, 0010, 0100, 1001, 0011,$$
$$0110, 1101, 1010, 0101, 1011, 0111, \ldots$$

with $F^{15}(k) = k$. Observe that, as expected from $(\star)$, the right-most bits $1, 1, 0, 0, 0, 1, \ldots$ are the keystream for 0111, starting from $k_3 = 1$.

(v) *Exercise:* Is every keystream generated by F a cyclic shift of the keystream for 0111? (For example, $x = 001101$ and $x' = 110100$ are equal up to cyclic shifts, since $x' = x_2x_3x_4x_5x_0x_1$.)

Lecture 14 In the cryptographic literature it is conventional to represent LFSRs by circuit diagrams, such as the one below showing F. By convention $\bigoplus$ denotes addition modulo 2, implemented in electronics by the XOR gate.



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

Cryptosystem defined by an LFSR.

Definition 5.3. 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 5.4. Define the decryption function $d_k : \mathbb{F}_2^n \to \mathbb{F}_2^n$.

Problem Sheet 5 shows how to encrypt an English message of length n by using the ASCII encoding to convert it to a word in $\mathbb{F}_2^{8n}$.

Invertible LFSRs and periods. **Lecture 15**

Exercise 5.5. Let H be the LFSR of width 4 with taps $\{1, 2\}$. Show that H is not invertible.

This exercise and Example 5.2(i) suggest the general result: an LFSR is invertible if and only if 0 is one of the taps. The steps in a proof are indicated in Question 3 of Sheet 4.

Exercise 5.6. Let G be the LFSR of width 4 with taps $\{0, 2\}$.

(a) Find the keystreams for the keys 0001 and 0011.
(b) Which words of length 4 do not appear in either keystream?
(c) Find all keystreams generated by this LFSR.

We saw in Example 5.2(v) a case where there was a unique keystream, up to cyclic shifts. The previous exercise shows that in general, there may be several different keystreams.

For cryptographic purposes, we want the keystream to be as long as possible before it repeats. The best possible, for an LFSR of width ℓ, is a single keystream that repeats only after $2^\ell - 1$ positions. As in Example 5.2(v), this keystream is then the unique non-zero keystream up to cyclic shifts.

Lemma 5.7. *Let F be an invertible LFSR of width ℓ.*

(i) *Let $k \in \mathbb{F}_2^\ell$. There exists $m \leq 2^\ell - 1$ such that $F^m(k) = k$.*
(ii) *There exists $m \in \mathbb{N}$ such that $F^m = \mathrm{id}$, the identity function.*

By this lemma the following definitions are well-defined.

Definition 5.8.

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

For example, the LFSRs F and G in Example 5.2 and Exercise 5.6 have periods 15 and 6, respectively. By Lemma 5.7, the period of a keystream of an LFSR of width ℓ is at most $2^\ell - 1$.

Lecture 16

The matrix representation of an LFSR. LFSRs are linear functions: if F is an LFSR of width ℓ then $F(x+x') = F(x) + F(x')$ for all x, $x' \in \mathbb{F}_2^\ell$. We can therefore represent each LFSR by a matrix.

Here it is most convenient to use row vectors: we say that an $\ell \times \ell$ matrix M *represents* an LFSR F of width ℓ if $xM = F(x)$ for all $x \in \mathbb{F}_2^\ell$. Here xM is the product of a row vector in $\mathbb{F}_2^\ell$ and the matrix M.

Proposition 5.9. *Let F be an LFSR of width ℓ and taps $T \subseteq \{0, 1, \ldots, \ell-1\}$. The matrix (acting on row vectors) representing F is*

$$\begin{pmatrix} 0 & 0 & 0 & \ldots & 0 & [0 \in T] \\ 1 & 0 & 0 & \ldots & 0 & [1 \in T] \\ 0 & 1 & 0 & \ldots & 0 & [2 \in T] \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \ldots & 0 & [\ell-2 \in T] \\ 0 & 0 & 0 & \ldots & 1 & [\ell-1 \in T] \end{pmatrix}$$

where

$$[t \in T] = \begin{cases} 1 & \textit{if } t \in T \\ 0 & \textit{otherwise.} \end{cases}$$

As a guide to the structure of this important matrix, some zero entries are printed in grey: this is just notation, and not of any mathematical significance.

The matrix in Proposition 5.9 is invertible if and only if $0 \in T$. So, as claimed earlier, an LFSR is invertible if and only if 0 is one of its taps.

Recall that the *minimal polynomial* of a matrix M with coefficients in $\mathbb{F}_2$ is the non-zero polynomial $g(X) \in \mathbb{F}_2[X]$ of least degree such that $g(M) = 0$.

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

In the following lemma we work with column vectors of length ℓ. For $i \in \{0, 1, \ldots, \ell-1\}$, let $\mathbf{v}(i)$ denote the column vector with 1 in position i (numbering positions from 0 as usual), and 0 in all other positions. The vectors $\mathbf{v}(0), \mathbf{v}(1), \ldots, \mathbf{v}(\ell-1)$ are shown in the margin.

Lemma 5.10. *Let F be an LFSR of width ℓ with taps T representing by the matrix M. Define $g(X) = X^\ell + \sum_{t \in T} X^t$.*

(a) *If $t < \ell$ then $M^t \mathbf{v}(0) = \mathbf{v}(t)$;*
(b) *$\sum_{t \in T} M^t \mathbf{v}(0) = M^\ell \mathbf{v}(0)$,*
(c) *$g(M)\mathbf{v} = 0$ for all column vectors $\mathbf{v}$,*
(d) *$g(X)$ is the minimal polynomial of M.*

Here $g(M)$ is obtained by substituting M for X, so $g(M) = M^\ell + \sum_{t \in T} M^t$. Recall that the minimal polynomial of a matrix M is the monic polynomial h of least degree such that $h(M) = 0$. Over $\mathbb{F}_2$, the only coefficients are 0 and 1, so the 'monic' condition always holds.

The following proof will be illustrated by an example in lectures. It is included for interest and logical completeness and is non-examinable.

Proof of Lemma 5.10. When $t = 0$ we have $M^t\mathbf{v}(0) = I\mathbf{v}(0) = \mathbf{v}(0)$. Suppose inductively that $M^{t-1}\mathbf{v}(0) = \mathbf{v}(t-1)$. Then

$$M^t\mathbf{v}(0) = M\big(M^{t-1}\mathbf{v}(0)\big) = M\mathbf{v}(t-1)$$

is column $t-1$ of M; if $t < \ell$ this is $\mathbf{v}(t)$, as required for (a).

Note that, by (a), $M^{\ell-1}\mathbf{v}(0) = \mathbf{v}(\ell-1)$. Hence $M^\ell\mathbf{v}(0) = M\mathbf{v}(\ell-1)$ is the rightmost column of M, namely $\sum_{t \in T}\mathbf{v}(t)$. By (a) this is $\sum_{t\in T} M^t\mathbf{v}(0)$, hence (b). By (b),

$$g(M)\mathbf{v}(0) = (M^\ell + \sum_{t\in T} M^t)\mathbf{v}(0) = \sum_{t\in T} M^t\mathbf{v}(0) + \sum_{t\in T} M^t\mathbf{v}(0) = \mathbf{0}.$$

More generally, if $s < \ell$ then, by (a) and this calculation,

$$g(M)\mathbf{v}(t) = g(M)M^t\mathbf{v}(0) = M^t g(M)\mathbf{v}(0) = M^t\mathbf{0} = \mathbf{0}.$$

Hence $g(M)\mathbf{v}(t) = 0$ for $0 \leq t < \ell$. Now let $\mathbf{v}$ be an arbitrary column vector with entries $b_0, \ldots, b_{\ell-1}$. We have

$$g(M)\mathbf{v} = g(M)\sum_{i=0}^{\ell-1} b_i\mathbf{v}(i) = \sum_{i=0}^{\ell-1} b_i g(M)\mathbf{v}(i) = \sum_{i=0}^{\ell-1} b_i\mathbf{0} = \mathbf{0}$$

proving (c).

Finally, by (c), $g(M) = 0$, the $\ell \times \ell$ zero matrix. Suppose $f(M) = 0$ where $f(X) = f_0 + f_1X + \cdots + f_dX^d$. If $d < \ell$ then $f(M)\mathbf{v}(0) = f_0\mathbf{v}(0) + \cdots + f_d\mathbf{v}(d) \neq \mathbf{0}$. Therefore g has the minimum possible degree ℓ and so is the minimal polynomial of M. $\square$

Motivated by the lemma we define the *minimal polynomial* of an LFSR F of width ℓ with taps T to be $g_F(X) = X^\ell + \sum_{t\in T} X^t$.

We now show the minimal polynomial determines the period. We need the following fact: if M is a matrix with entries in $\mathbb{F}_2$ and $h(X) \in \mathbb{F}_2[X]$ is a polynomial such that $h(M) = 0$ then $h(X)$ is divisible by the minimal polynomial of M.[10]

[10] *Proof:* let $g(X)$ be the minimal polynomial of M. By polynomial division we have $h(X) = q(X)g(X) + r(X)$ where either $r(X) = 0$ or $\deg r(X) < \deg g(X)$. Then $0 = h(M) = q(M)g(M) + r(M) = q(M)0 + r(M) = r(M)$ so $r(M) = 0$. But $g(X)$ has the least degree of non-zero polynomials such that $g(M) = 0$, so $r(X) = 0$, i.e. $g(X)$ divides $h(X)$.

Corollary 5.11. *The period of an invertible LFSR F is the least m such that $g_F(X)$ divides $X^m + 1$.*

Proof. We first prove that

$$g_F(X) \text{ divides } X^d + 1 \iff F^d = \text{id}.$$

Suppose that $g_F(X)$ divides $X^d + 1$. Then $g_F(X)h(X) = X^d + 1$ for some polynomial $h(X)$ and so $0 = g_F(M)h(M) = M^d + I$. Hence $M^d = I$ and so $F^d = \text{id}$. Conversely, if $F^d = \text{id}$ then $M^d = I$ and so by the fact above, $X^d + 1$ is divisible by $g_F(X)$. The corollary now follows from the definition of period. □

It is a useful fact that every LFSR has a cycle of length equal to its period. (For a proof, non-examinable, see the optional Question 5 on Sheet 4.) Since there are $2^\ell - 1$ non-zero elements of $\mathbb{F}_2^\ell$, this implies that the period of an LFSR of width ℓ is at most $2^\ell - 1$.

To work with Corollary 5.11, the following lemma is useful. Let $\text{hcf}(d, e)$ denote the highest common factor of $d, e \in \mathbb{N}$.

Lemma 5.12. *If a polynomial $g(X)$ divides $X^d + 1$ and $X^e + 1$ then it divides $X^{\text{hcf}(d,e)} + 1$.*

Example 5.13. The number $2^{13} - 1 = 8191$ is a prime. The MATHEMATICA command `Factor[X^8191 + 1, Modulus -> 2]` returns

$$(1 + X)(1 + X + X^3 + X^4 + X^{13})(1 + X + X^2 + X^5 + X^{13})\ldots.$$

(Here ... stands for 630 omitted factors all of degree 13.) The taps of the LFSR of width 13 with minimal polynomial $1 + X + X^3 + X^4 + X^{13}$ are $\{0, 1, 3, 4\}$. By Corollary 5.11, its period is the least m such that $1 + X + X^3 + X^4 + X^{13}$ divides $X^m + 1$. If $1 + X + X^3 + X^4 + X^{13}$ divides $X^e + 1$ with $e < 8191$ then, by Lemma 5.12, $1 + X + X^3 + X^4 + X^{13}$ divides $X^{\text{hcf}(e,8191)} + 1 = X + 1$, a contradiction. Therefore[11] the period is 8191.

Primes such as $2^{13} - 1$ are known as *Mersenne primes*. The largest known prime number is the Mersenne prime $2^{277\,232\,917} - 1$ found by the Great Internet Mersenne Prime Search.

6. PSEUDO-RANDOM NUMBER GENERATION

Lecture 17 The keystream generated by an invertible LFSR can be used as a source of random numbers. In this section we look at its statistical properties.

[11]Using some field theory one can see this in another way. If $g(X) \in \mathbb{F}_2[X]$ is an irreducible polynomial of degree 13 then g splits in the finite field $\mathbb{F}_{2^{13}}$. Since $|\mathbb{F}_{2^{13}}^\times| = 8191$, and 8191 is prime, all the roots of g have order 8191. Therefore g is a primitive polynomial and the corresponding LFSR has maximum possible period 8191. Moreover, all $(8191 - 1)/13 = 630$ irreducible polynomials of degree 13 are obtained by factorizing $X^{8191} + 1$, as seen in the MATHEMATICA calculation.

By Lemma 5.7(i), that the maximum possible period of a keystream of an LFSR of width ℓ is $2^\ell - 1$. Such an LFSR has period $2^\ell - 1$. 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.)

Exercise 6.1. Let F be the LFSR of width 4 with taps $\{0, 1\}$ and period $15 = 2^4 - 1$ seen in Example 5.1. It has the maximum possible period for its width. The keystream for $k = (1, 1, 0, 0)$ 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,

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

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

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

(a) Find all the positions t such that

$$(k_t, k_{t+1}, k_{t+2}, k_{t+3}) = (0, 1, 1, 1).$$

(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 t such that $(k_t, k_{t+1}, k_{t+2}) = (0, 1, 1)$. How many are there?

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

Proposition 6.2. *Let F be an invertible LFSR of width ℓ and 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 positions t within this cycle, so $0 \leq t < 2^\ell - 1$.*

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

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

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

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

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? **Lecture 18**

Exercise 6.3. 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, 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?

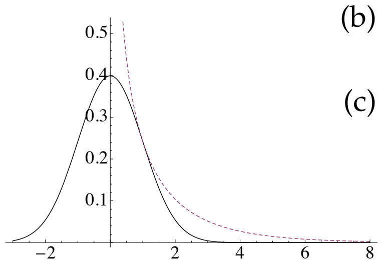
Here we answer this question for the first test in Exercise 6.3, counting the number of zeros and ones. This is the *monobit test*.

Exercise 6.4. 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 $n = 100$ has 60 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.]



See Question 4 on Problem Sheet 5 for the analogous test looking at pairs.

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

Lecture 19

Definition 6.5. Given $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1}) \in \mathbb{F}_2^n$ define

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

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

Exercise 6.6. Find the correlation between a generating cycle for the LFSR of width 3 with taps $\{0, 1\}$ and each cyclic shift of itself. Does your answer depend on the key?

More generally we shall prove the following proposition.

Proposition 6.7. *Let $(k_0, k_1, \ldots, k_{2^\ell-2})$ be a generating cycle of a maximal period LFSR of width ℓ. The correlation between $(k_0, k_1, \ldots, k_{2^\ell-2})$ and any proper cyclic shift of $(k_0, k_1, \ldots, k_{2^\ell-2})$ is $-1/(2^\ell - 1)$.*

Again this shows that a generating cycle of an LFSR of maximum possible period for its width has a strong randomness property.

7. Non-linear stream ciphers

Lecture 20

A general stream cipher takes a key $k \in \mathbb{F}_2^\ell$, for some fixed ℓ, and outputs a sequence $u_0, u_1, u_2, \ldots$ of bits. For each $n \in \mathbb{N}$ there is a corresponding cryptosystem where, as in Definition 5.3, 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 7.1. In the LFSR cryptosystem of Definition 5.3, the sequence $u_0, u_1, u_2, \ldots$ is simply the keystream $k_0, k_1, k_2, \ldots$. 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 7.2. One 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 $\{0, 3\}$ of period 15.

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

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

Unfortunately, $(u_0, u_1, u_2, \ldots)$ is also generated by F: it is the keystream for $(1, 0, 0, 1)$. *Exercise:*

(a) Explain why this should have been expected. [*Hint:* the same linearity was used to prove Proposition 6.7.]

(b) *Exercise:* can the keys k and $k^\star$ be recovered from $(u_0, u_1, \ldots, u_{19})$? If so, explain how; if not, is this a problem for the known plaintext attack?

- Let F' be the LFSR of width 3 with taps $\{0, 1\}$ of period 7.

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

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

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 linear algebra method from Question 3 on Sheet 5 shows that the first 10 bits of $(u_0, u_1, u_2, \ldots)$ are generated by the LFSR of width 7 with taps $\{0, 1, 5, 6\}$. In fact this holds for the entire sequence.[12] So as in (b) above, $(u_0, u_1, u_2, \ldots)$ is the output of a single LFSR, and so the cryptosystem is weak.

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 7.3. 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 $(k_0, k_1, k_2, \ldots)$ and $(k'_0, k'_1, k'_2, \ldots)$ be keystreams for F and F'
- Let $(g_0, g_1, g_2, \ldots)$ be a keystream for G.

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

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

Lecture 21

For example, if F is the LFSR of width 3 with taps $\{0, 1\}$, F' is the LFSR of width 4 with taps $\{0, 3\}$, and G is the LFSR of width 4 with taps $\{0, 1\}$ and $(g_0, g_1, g_2, g_3) = (0, 0, 0, 1)$ then **[corrected after lecture: F and F' got swapped by mistake]**

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

Exercise: 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 $(k_0, k_1, k_2, \ldots)$ and $(k'_0, k'_1, k'_2, \ldots)$, or their product (see Question 1 on Sheet 6). But it is vulnerable to a correlation attack.

[12] See the optional Question 4 on Sheet 6 for a proof this happens regardless of the choice of the keys k and k'.

Exercise: Assume the keys $(k_0, k_1, \ldots, k_{\ell-1})$ and $(k'_0, k'_1, \ldots, k'_{\ell'-1})$ are chosen with equal probability from $\mathbb{F}_2^{\ell}$ and $\mathbb{F}_2^{\ell'}$, respectively. Find $\mathbf{P}[k_s = u_s]$ for each $s \in \mathbb{N}_0$.[13]

Thus the correlation between $(k_0, k_1, k_2, \ldots)$ and $(u_0, u_1, u_2, \ldots)$ is $\frac{3}{4} - \frac{1}{4} = \frac{1}{2}$. Recall that 0 corresponds to no correlation, 1 to equality in every position and -1 to inequality in every position.

Attack 7.4. *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 $(v_0, v_1, \ldots, v_{n-1})$ and $(u_0, u_1, \ldots, u_{n-1})$. If the correlation is not nearly $\frac{1}{2}$ then the candidate key is rejected. Otherwise it is likely that $(k_0, \ldots, k_{\ell-1}) = (v_0, \ldots, v_{\ell-1})$.*

Exercise: is it better to guess the key for F or for F'?

One can repeat Attack 7.4 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 $2^{\ell+\ell'}$ guesses required by trying every possible pair of keys. (Question 1(b) on Sheet 6 suggests some ways to speed up the second step for k'.)

An attack such as Attack 7.4 is said to be *sub-exhaustive* because it finds the key using fewer guesses than brute-force exhaustive search through the keyspace.

The Geffe cipher is weak because each keystream bit xy is a product biased to 0. Adding up multiple bits reduces this effect.

Example 7.5. Let F be the LFSR of width 5 with taps $\{0, 2\}$ and let F' be the LFSR of width 6 with taps $\{0, 1, 3, 4\}$. 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

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

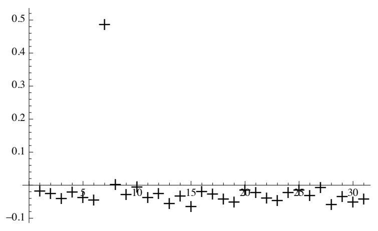
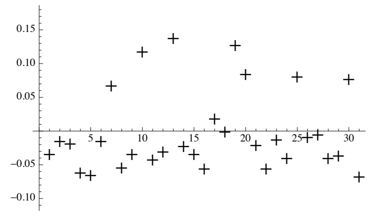
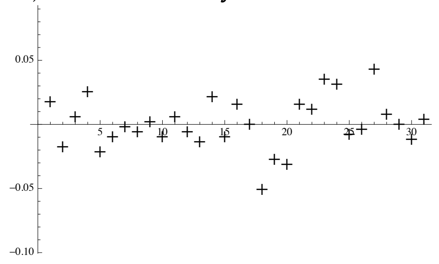
Note that there are m products in the sum. Define $u_s = 0$ if $0 \leq s < m - 1$. [**Corrected from** $s < m$] The *m-quadratic stream cipher* is the cryptosystem defined using the sequence $u_0, u_1, \ldots, u_{1023}$.

Taking $m = 1$ gives a cipher like the Geffe generator: since $u_s = k_s k'_s$ we have $\mathbf{P}[u_s = k_s] = \frac{3}{4}$, giving a correlation of $\frac{1}{2}$. Attack 7.4 is effective.

For general m, the expected correlation between keystream of the m-quadratic stream cipher $u_0 u_1 u_2 \ldots u_{1023}$ and the keystream $k_0 k_1 k_2 \ldots k_{1023}$ of the LFSR of width 5 is about $\frac{1}{2^m}$. (If time permits this will be proved 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.

[13]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.

The graphs below show correlations for all 31 non-zero keys k when $m = 1$, $m = 3$ and $m = 5$. The correct key is $(0, 0, 1, 1, 1)$, or 7 in binary.

Exercise 7.6. Unfortunately the m-quadratic cipher is still vulnerable because taking 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 which, like the quadratic cipher mixes multiplication and addition on multiple LFSRs. This combination gives a practical cipher with no known sub-exhaustive attacks.

Example 7.7 (TRIVIUM)**.** Take three LFSRs of widths 93, 84 and 101, tapping positions $\{0, 27\}$, $\{0, 15\}$ and $\{0, 45\}$, with internal states $x \in \mathbb{F}_2^{93}$, $x' \in \mathbb{F}_2^{84}$, $x'' \in \mathbb{F}_2^{101}$. The keystream is defined by

$$k_s = x_0 + x_{27} + x'_0 + x'_{15} + x''_0 + x''_{45}.$$

The feedback functions are

$$\begin{aligned} f\big((x_0, \dots, x_{92})\big) &= x_0 + x_{27} + x_1 x_2 + x'_6 \\ f'\big((x'_0, \dots, x'_{84})\big) &= x'_0 + x_{15} + x'_1 x'_2 + x''_{24} \\ f''\big((x''_0, \dots, x''_{101}\big) &= x''_0 + x''_{14} + x''_1 x''_2 + x_{24} \end{aligned}$$

In each case the final summand introduces a bit from a different shift register.

Rather than use a 288-bit key, TRIVIUM uses a (secret) 80-bit key, and a (non-secret) 80-bit initialization vector, setting the other positions in the internal state to 0.[14] The first 1152 bits of the keystream are unusually biased, and so are discarded.

[14]See `http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf` for details: for consistency, the right-shifting registers in the formal specification have been converted to (equivalent) left-shifting registers.

(C) Block ciphers

8. Introduction to block ciphers and Feistel networks

Lecture 22

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(ii), 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 128 bits, they have to be split into multiple *blocks*, each of n bits, before encryption.

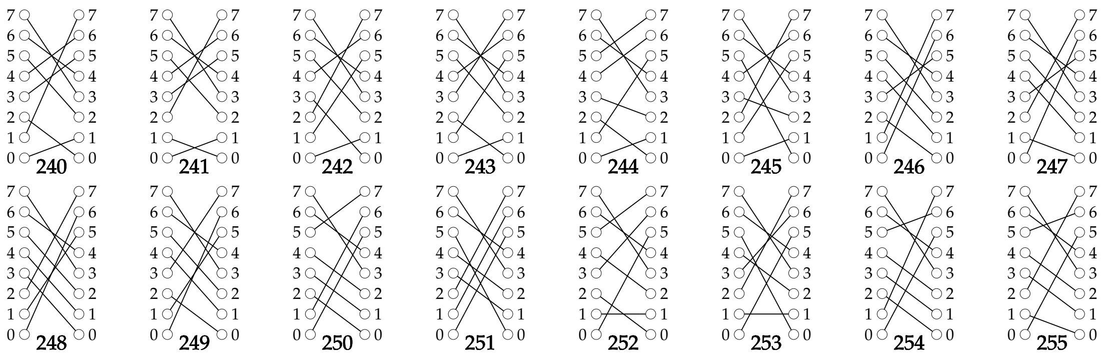
Example 8.1. The binary one-time pad of length n is the block cipher of block size n and key length n in which $e_k(x) = x + k$ for all $k \in \mathbb{F}_2^n$.

The one-time pad has perfect secrecy (see Question 3 on Sheet 3). 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 8.2. 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 permutations $e_k : \mathbb{F}_2^3 \to \mathbb{F}_2^3$ for $k \in \mathcal{K}$, chosen according to a fairly arbitrary rule (details omitted). For example, since $11111100 \in \mathbb{F}_2^8$ is the binary form of **252**, and $010 \in \mathbb{F}_2^3$ is the binary form of 2, diagram **252** shows that $e_{11111100}(010) = 000$.



The other 240 permutations are posted on Moodle and will be available in the lecture.

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

(i) By a chosen plaintext attack Mark learns that $e_k(000) = 101$ and $e_k(001) = 111$. One possible key is 254. There are six others: find at least one of them.
(ii) By choosing two further plaintexts Mark learns that $e_k(011) = 001$ and $e_k(110) = 011$. Determine k.
(iii) Later Mark's boss Eve observes the ciphertext 100. What is $d_k(100)$?

In this case since $|\mathbb{F}_2^3| = 8$, there are $8! = 40320$ permutations of $\mathbb{F}_2^3$, of which 256 were used.

A typical modern block cipher uses $2^{128} \approx 3.40 \times 10^{30}$ **[corrected after lecture]** permutations of $\mathbb{F}_2^{128}$. To store *just one* of these permutations 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. Experience shows that composing sufficiently many simple operations gives encryption functions that 'look random'.

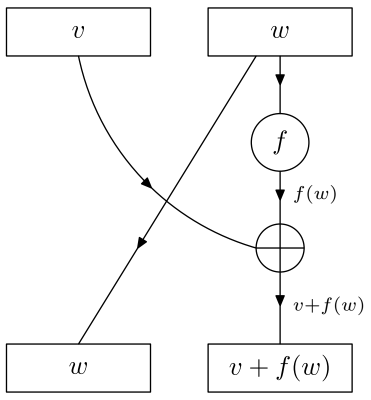
Feistel networks.

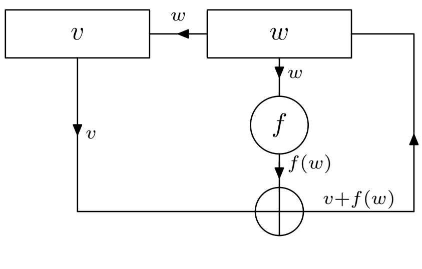
Definition 8.3. Let $m \in \mathbb{N}$ and let $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$ be a function. Given v, $w \in \mathbb{F}_2^m$, let (v, w) denote $(v_0, \ldots, v_{m-1}, w_0, \ldots, w_{m-1}) \in \mathbb{F}_2^{2m}$. The *Feistel network* for f is the function $F : \mathbb{F}_2^{2m} \to \mathbb{F}_2^{2m}$ 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.

The circuit diagrams below show two equivalent definitions of the Feistel network: the right-hand diagram makes the analogy with LFSRs more obvious.





Exercise 8.4. Show that, for any function $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$, the Feistel network F for f is invertible. Give a formula for its inverse in terms of f.

See Question 3 on Problem Sheet 7 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 8.5 (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 contents of the right register shift to the left, 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 8.6.

(a) Suppose that $k = 0001\,0011\,0000$, shown split into the three round keys. Show that **Lecture 23**

$$e_k\big((0, 0, 0, 0, 0, 0, 0, 0)\big) = (1, 1, 1, 0, 1, 1, 0, 1)$$

(b) Find $d_k\big((0, 0, 0, 0, 0, 0, 0, 0)\big)$ if the key is as in (a).

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

Exercise 8.7. Suppose we change the Feistel function in round i to $x \mapsto S(x) + k^{(i)}$. What is $(v^{(1)}, v^{(2)})$ in terms of v, w and $k^{(1)}$? Which cipher is likely to be stronger?

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 permutation (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.

No sub-exhaustive attacks on DES are known. But the relatively small keyspace $\mathbb{F}_2^{56}$ means that exhaustive attacks are practical. Therefore DES cannot be considered secure. Some timings:

- 1997: 140 days, distributed search on internet
- 1998: 9 days 'DES cracker' (special purpose) $250000
- 2017: 6 days 'COPACOBONA' (35 FPGA's) $10000

Lecture 24

Exercise 8.8. Suppose we apply 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 $(k, k') \in \mathbb{F}_2^{56} \times \mathbb{F}_2^{56}$,

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

(a) How long would a brute force exhaustive search over $\mathbb{F}_2^{56} \times \mathbb{F}_2^{56}$ take?
(b) Does this mean 2DES is secure?

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

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

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.

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. The non-linearity comes from a function defined using the finite field $\mathbb{F}_{2^8}$.

The next two examples show the two main building blocks of AES.

Example 8.9. 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 defined by

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

We will define the decryption functions in lectures and see that, used on its own, the affine block cipher is vulnerable to a chosen plaintext attack.

Lecture 25

Example 8.10. Let α be an indeterminate. Define

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

Elements of $\mathbb{F}_2^8$ are added and multiplied like polynomials in α, but whenever you see a power α^d where $d \geq 8$, eliminate it[16] using the rule

$$1 + \alpha + \alpha^3 + \alpha^4 + \alpha^8 = 0.$$

For example $(1 + \alpha) + (\alpha + \alpha^5) = 1 + \alpha^5$ and

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

Multiplying the defining rule for α by α^{-1}, we get $\alpha^{-1} + 1 + \alpha^2 + \alpha^3 + \alpha^7 = 0$ so $\alpha^{-1} = 1 + \alpha^2 + \alpha^3 + \alpha^7$.

Definition 8.11. Let $\mathbb{F}_{2^8}$ be the finite field of size 2^8 as in Example 8.10. Define $p : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ by

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

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

For example, 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 $P(1000\,0000) = 10000000$
(2) $0100\,0000 \longleftrightarrow \alpha \in \mathbb{F}_{2^8}$ and $\alpha^{-1} = 1 + \alpha^2 + \alpha^3 + \alpha^7$ was found in Example 8.10, so $P(0100\,0000) = 10110001$.

Exercise: Show that $P(0010\,0000) = 1101\,0011$.

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

Lecture 26

[16]An equivalent definition using some ring theory is $\mathbb{F}_{2^8} = \mathbb{F}_2[X]/\langle 1 + X + X^3 + X^4 + X^8\rangle$; now α is the coset $X + \langle 1 + X + X^3 + X^4 + X^8\rangle$ in the quotient ring. The polynomial $1 + X + X^3 + X^4 + X^8$ was chosen by the designers of AES: it is irreducible (this is essential) but not primitive.

- The pseudo inverse function $P : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ is applied to each subblock followed by an affine transformation $\mathbb{F}_2^8 \to \mathbb{F}_2^8$, of the type in Example 8.9. This gives confusion and diffusion *within each subblock*. (SUBBYTES.)
- Diffusion across all 128 bits comes from a row permutation of the 16 subblocks, organized into a 4×4 grid

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

  and a further mixing of each column by an invertible linear map (SHIFTROWS and MIXCOLUMNS)
- The round key in $\mathbb{F}_2^{128}$ is added (ADDROUNDKEY).

There are no known sub-exhaustive attacks on AES. It is the most commonly used block cipher. Since 2010 Intel and AMD microprocessors have supported AES as a primitive operation. AES was defined to be efficient in hardware: for example, the subblocks fit exactly into 8-bit bytes.

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

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 x is longer than n bits, it must be split into blocks $x^{(1)}, \dots, x^{(m)} \in \mathbb{F}_2^n$:

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

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:

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

- 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.

9. Differential cryptanalysis

Differential cryptanalysis was known to the designers of DES in 1974. They kept it secret, at the request of the NSA. It was rediscovered in the late 1980s.

One important idea is seen in the attack on the reused one-time pad in Question 2 on Problem Sheet 3. We have unknown plaintexts $x, x_\Delta \in \mathbb{F}_2^n$, an unknown key $k_{\mathrm{otp}} \in \mathbb{F}_2^n$, and known ciphertexts $x + k_{\mathrm{otp}}$ and $x_\Delta + k_{\mathrm{otp}}$. Adding the known ciphertexts gives $x + x_\Delta$, independent of k_{otp}.

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

The DES S-boxes and the pseudo-inverse function $P : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ in AES are chosen to avoid this weakness. By the exercise below an output difference of 1 to P can come from many different input differences.

Exercise 9.1. Let $\Gamma \in \mathbb{F}_2^8$ be non-zero. Show that

$$\{w \in \mathbb{F}_2^8 : P(w) + P(w + \Gamma) = 1\}$$

has size 0 or 2, except when $\Gamma = 1$, when it has size 4. [*Hint:* quadratic equations over any field have at most two roots.]

A similar result holds replacing 1 with a general non-zero $\Delta \in \mathbb{F}_2^8$; the exceptional case is then $\Gamma = \Delta^{-1}$. (See the optional question on Problem Sheet 8.)

Exercise 9.2. Fix $\Gamma = \mathbb{F}_2^8$. Let $w \in \mathbb{F}_2^8$ be chosen uniformly at random. What are the possible values for $\mathbf{P}[P(w) + P(w + \Gamma) = 1]$?

Lecture 27

Attack 9.3. *Let $e_k : \mathbb{F}_2^n \to \mathbb{F}_2^n$ for $k \in \mathbb{F}_2^\ell$ be the encryption functions for a block cipher of block size n and key length ℓ. For $(k_{otp}, k) \in \mathbb{F}_2^n \times \mathbb{F}_2^\ell$, define $E_{(k_{otp},k)} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ by*

$$E_{(k_{otp},k)}(x) = e_k(x + k).$$

Let $\Delta \in \mathbb{F}_2^n$. In a chosen plaintext attack on the cryptosystem E, we choose $x \in \mathbb{F}_2^n$ and a difference $\Delta \in \mathbb{F}_2^n$ and obtain the ciphertexts

$$z = E_{(k_{otp},k)}(x)$$
$$z_\Delta = E_{(k_{otp},k)}(x + \Delta)$$

Set $\Gamma = z + z_\Delta$. Then $e_k^{-1}(z) + e_k^{-1}(z_\Delta) = \Delta$. Moreover, for $k_{guess} \in \mathbb{F}_2^\ell$, either

$$e_{k_{guess}}^{-1}(z) + e_{k_{guess}}^{-1}(z_\Delta) \neq \Delta$$

and we deduce $k_{guess} \neq k$, or

$$e_{k_{guess}}^{-1}(z) + e_{k_{guess}}^{-1}(z_\Delta) = \Delta$$

and then $k_{guess} \in \mathcal{K}_z$ where

$$\mathcal{K}_z = \big\{k_{guess} \in \mathbb{F}_2^n : e^{-1}_{k_{guess}}(z) + \mathrm{e}^{-1}_{k_{guess}}(z+\Gamma) = \Delta\big\}.$$

[There was an erroneous exercise here: Exercise 9.5 is what was meant.]

Intuitively: for the correct key k, undoing the second cipher we get back the difference Δ; for wrong keys, we get Δ only if k_{guess} has the special property that $k_{\text{guess}} \in \mathcal{K}_z$, where $z = E_{(k_{\text{otp}},k)}(x)$.

If the block cipher is good then $\mathcal{K}_z$ is small. Therefore *false keys*, where we do not immediately see that our guess is wrong, are rare. Note that we guess k, but not k_{otp}.

Attack on the AES S-box. We apply Attack 9.3 to a cryptosystem based on the pseudo-inverse function $P : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ used in AES.

Example 9.4. Take $n = 8$ and $\ell = 8$. For $k \in \mathbb{F}_2^8$, define

$$e_k(y) = P(y) + k$$

Note that $e_k^{-1}(z) = P(z+k)$ and so

$$e^{-1}_{k_{\text{guess}}}(z) + e^{-1}_{k_{\text{guess}}}(z_\Delta) = P(z + k_{\text{guess}}) + P(z_\Delta + k_{\text{guess}}).$$

By definition $z_\Delta = z + \Gamma$. Hence the set $\mathcal{K}_z$ in Attack 9.3 is

$$\mathcal{K}_z = \{k_{\text{guess}} \in \mathbb{F}_2^8 : P(z + k_{\text{guess}}) + P(z + k_{\text{guess}} + \Gamma) = \Delta\}.$$

Running the attack: Take $\Delta = 1000\,0000$; this corresponds to $1 \in \mathbb{F}_{2^8}$. For each $k_{\text{guess}} \in \mathbb{F}_2^8$, we compute $P(z + k_{\text{guess}}) + P(z_\Delta + k_{\text{guess}})$. If the answer is Δ then $k_{\text{guess}} \in \mathcal{K}_z$ and k_{guess} is either k or a false key. Otherwise we reject k_{guess}.

By the generalized version of Exercise 9.1, there are usually exactly two different $k_{\text{guess}} \in \mathbb{F}_2^8$ such that $P(z + k_{\text{guess}}) + P(z + k_{\text{guess}} + \Gamma) = \Delta$. One must be k.

Exercise 9.5. Show that $k + \Gamma \in \mathcal{K}_z$

So usually $\mathcal{K}_z = \{k, k + \Gamma\}$ and the attack in Attack 9.3 finds the key and the false key $k + \Gamma$; very rarely, when $\Delta = \Gamma^{-1}$, there are three false keys.

In the following examples we take $k_{\text{otp}} = 0000\,0000$.

(1) If $k = 0000\,0000$ and $x = 0100\,0000$ then, since $P(0100\,0000) = 1011\,0001$ and $P(1100\,0000) = 0110\,1111$, $z + z_\Delta = 1101\,1110$. There are exactly 2 keys k_{guess} such that $k \in \mathcal{K}_z$, namely

$$0000\,0000,\ 1101\,1110.$$

(2) If $k = 0000\,0000$ and $x = 0000\,0000$ then $z + z_\Delta = 1000\,0000$ and there are exactly 4 keys k_{guess} such that $k \in \mathcal{K}_z$, namely

$$0000\,0000,\ 1000\,0000,\ 0011\,1101,\ 1011\,1101.$$

(To check this you will need to know $P(0011\,1101) = 1011\,1101$ and so, since $P(P(x)) = x$ for all $x \in \mathbb{F}_2^8$, $P(1011\,1101) = 0011\,1101$.) This is the exceptional case when $\Delta^{-1} = \Gamma$.

(3) *Exercise:* let $k = 1111\,1111$. What are the guesses k_{guess} if $x = 0100\,0000$? What if $x = 0000\,0000$? [*Hint:* use (1) and (2).]

Exercise 9.6.

(a) Show that the attack typically finds k and the false key $k + \Gamma$ using at most 2×2^8 decryptions to calculate $e^{-1}_{k_{\mathrm{guess}}}(z)$ and $e^{-1}_{k_{\mathrm{guess}}}(z_\Delta)$.

(b) How many encryptions are needed to test all the pairs (k_{otp}, k) and $(k_{\mathrm{otp}}, k + \Gamma)$ for $k_{\mathrm{otp}} \in \mathbb{F}_2^8$?

(c) Deduce that the attack finds the key (k_{otp}, k) using at most 2^{10} decryptions/encryptions. Why is this sub-exhaustive?

Example 9.4 should be compared with the meet-in-the-middle attack on 2DES: both show that composing two block ciphers may not give a significantly stronger cipher.

Attack on the Q-block cipher. Recall that we 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 8.5), the Feistel network sends (v, w) to $\big(w, v + S(w + k^{(1)})\big)$ where **Lecture 28**

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

Lemma 9.7.

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

(ii) *For any $(v, w) \in \mathbb{F}_2^8$ and any round key $k^{(1)} \in \mathbb{F}_2^4$ we have*

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

By (ii) the first round of the Q-block cipher has a similar weakness to the one-time pad: plaintexts (v, w) differing by $(0000, 1000)$ are encrypted to vectors differing by $(1000, 0010)$. Thus the first round of the Q-block cipher behaves like a one-time pad, *provided* we take $\Delta = (0000, 1000)$.

Example 9.8. We run Attack 9.3 on the Q-block cipher by taking $\Delta = (0000, 1000)$ and guessing the final 8 bits of the key k to undo the final two rounds.

Take $k = 0000\,0000\,0000$ and $x = 0000\,0001$. There are 16 keys $k_{\mathrm{guess}} \in \mathbb{F}_2^8$ such that $k_{\mathrm{guess}} \in \mathcal{K}_z$, namely all binary words of the form $\star 0 \star 0\ \star 0 \star 0$. These are the possibilities for

$$(k^{(2)}_{\mathrm{guess}}, k^{(3)}_{\mathrm{guess}}) \in \mathbb{F}_2^8.$$

Trying each guess together with all 16 possibilities for $k^{(1)}_{\mathrm{guess}} \in F_2^4$ we get

$$k \in \{0000\,0000\,0000,\ 1000\,0010\,1000,\ 1110\,1000\,0010,\ 0110\,1010\,1010\}.$$

All these keys encrypt 0000 0001 to the same ciphertext, namely 0000 0100. Repeating the attack with a different plaintext shows that k is one of the first two keys.

That we are left with two keys is explained by Question 2 on Sheet 8: it follows from Lemma 9.7(i) that, in the Q-block cipher, the encryption functions e_k and $e_{k+1000\,0010\,1000}$ are the same.

(D) Public key ciphers and digital signatures

10. Introduction to public key cryptography

We begin with a way that Alice and Bob can establish a shared secret key, communicating only over the insecure channel on page 4.

Everything in red is private. Everything not in red is known to the whole world— this includes the eavesdropper Eve. (This is not a standard convention, and you are welcome to ignore it if you prefer.)

Example 10.1. 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 and 2^b are sent publically.)
(3) Alice computes $(2^b)^a \bmod p$ and Bob computes $(2^a)^b \bmod p$.
(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 10.2. 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 $2^8 \equiv 7 \times 2 = 14 \equiv 3 \bmod 11$.

This exercise shows two number-theoretic facts that will be needed below. (See also Fact 10.5 below.)

- Fermat's Little Theorem: $c^{p-1} \equiv 1 \bmod p$ **[Corrected from mod** c**]** for any c not divisible by p.

- If $c^m \not\equiv 1 \bmod p$ for all m such that $1 \leq m < p-1$, then c is said to be a *primitive root modulo p*. If c is a primitive root then, working modulo p, we have
$$\{1, c, c^2, \ldots, c^{p-2}\} = \{1, 2, \ldots, p-1\}$$
Primitive roots always exist[17]: in Exercise 10.2 we took $c = 2$.

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 10.1, modified to avoid some potential weaknesses, and implemented efficiently.

- 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.)
- Alice and Bob compute $g^a \bmod p$ and $g^b \bmod p$ by repeated squaring: see Question 1 on Sheet 9 [**Not Sheet 7**]. This method is faster than the repeated doubling seen in Exercise 10.2. 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 [**Not 2**] 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.

[17]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:* if an abelian group A has elements of order t and t' then it has an element of order $\text{lcm}(t, t')$. Hence if t is greatest such that $\mathbb{Z}_p^\times$ has an element of order t then $x^t = 1$ for all $x \in \mathbb{Z}_p^\times$. But a polynomial of degree t has at most t roots, hence $t \geq p-1$. $\square$

Diffie–Hellman key exchange is secure only if, given g and g^x 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.

Exercise 10.3. Why do we exclude $p-1$ from the domain of f?

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 10.4. *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 $c = 5$ 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 10.4, 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 10.5. *Let p and q be distinct primes. Let $n = pq$. If*

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

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

Example 10.6. Let $p = 11$, $q = 17$, so $n = pq = 187$ and $(p-1)(q-1) = 160$. Let $c = 9$. Adapting the proof for Lemma 10.4, 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 c, it is easy to find r as in Fact 10.5. But it is believed to be hard to find r given only n and c. If so, $x \mapsto x^c \bmod n$ 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

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

The *public key* corresponding to (p, q, c) is (n, c) and the *private key* corresponding to (p, q, c) is (n, r), where $cr \equiv 1 \bmod (p-1)(q-1)$. The encryption function for (p, q, c) is

$$x \mapsto x^c \bmod n$$

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, c)) can decrypt.

Example 10.7.

(1) For a small example, take p and q as in Example 10.6. 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 $x^c \bmod n$ 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 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.

Extra: Historical note. Diffie–Hellman Key Exchange was published[18] in 1976. The RSA Cryptosystem, named after Rivest, Shamir and Adleman was published[19] 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.

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 $\mathbb{F}_2^2$.

Quantum computers can perform some computations far more quickly than classical computers. In particular, using Shor's Algorithm, a quantum computer can quickly factor integers. 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.) 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.

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

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

11. Digital signatures and hash functions

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

Digital signatures. Suppose Alice and Bob have RSA keys:

	public	private
Alice	(m, a)	(m, r)
Bob	(n, b)	(n, s)

Suppose Bob wants to tell Alice his bank details in a message x. He looks up her public key (m, a) and sends her $x^a \bmod m$. (Assume that $x < m$.)

Malcolm, the man-in-the-middle, cannot decrypt $x^a \bmod m$, because he does not know r. But if he has control of the channel, he can replace $x^a \bmod m$ with another $x'^a \bmod m$, of his choice.

To do this, Malcolm must 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 Malcolm cannot attack in this way.

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

Example 11.1. Alice is expecting a message from Bob. She receives z, and computes $d_a(z) = z^r \bmod m$, but gets garbage. Thinking that Bob has somehow confused the keys, she computes $z^b \bmod n$, and gets the ASCII encoding of

'Bob here, my account number is 40081234'.

(a) How did Bob compute z?
(b) Should Alice believe z was sent by Bob?
(c) Can Malcolm read z?
(d) How can Bob avoid the problem in (c)?

Let $x \in \mathbb{N}_0$ be Bob's message. If Bob's RSA number n is about 2^{2048} then the message x is a legitimate ciphertext only if $x < 2^{2048}$. This may seem big, but, using the 7-bit ASCII coding, it means only $2048/7 \approx 290$ characters can be sent. Bob can get round this by splitting the message into blocks, but computing $d_b(x^{(i)})$ for each block $x^{(i)} \in \{1, \ldots, n-1\}$ is slow. It is better to send x, and then append $d_b(v)$ where v is a hash of x.

Hash functions and the birthday paradox.

Definition 11.2.

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

(b) Let (n, b) be Bob's public key in the RSA cryptosystem. The pair $\big(x, d_b(h(x))\big)$ is a *signed message* from Bob.

Alice *verifies* that a pair (x, s) is a valid signed message from Bob by checking that $h(x) = e_b(s)$.

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 $x' \in \mathbb{N}$ such that $x' \neq x$ and $h(x') = h(x)$. (*Preimage resistance.*)

(c) It is hard to find a pair (x, x') with $x \neq x'$ such that $h(x) = h(x')$. (*Collision resistance.*)

When Alice receives the signed message (x, s) from Bob, she verifies that $h(x) = e_b(s)$, and so $s = d_b(h(x))$. She now knows that Bob has decrypted (that is signed), the hash value $h(x)$. Only Bob 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 Alice can be confident that x really is Bob's message.

A good hash function of length m behaves like a random function from $\mathbb{N}_0$ to $\mathbb{F}_2^m$. Given a hash value $v = h(x)$, a brute-force search for x' such that $h(x') = v$ will succeed on each x' with probability $\frac{1}{2^m}$. Hence the number of trials until first success is distributed geometrically with parameter $\frac{1}{2^m}$, so on average 2^m trials are needed. Thus in (b) 'hard to find' means 'requires at least 2^m hashes'.

Exercise 11.3. Let $h : \mathbb{N} \to \mathbb{F}_2^m$ be a good hash function. On average, how many hashes does an attacker need to calculate to find a pair (x, x') with $h(x) = h(x')$?

The mathematics behind Exercise 11.3 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.

Lemma 11.4. *If there are B possible birthdays then in a room of $\sqrt{2 \ln 2}\sqrt{B}$ people, the probability is about $\frac{1}{2}$ that two people have the same birthday.*

For instance, when $B = 365$, Lemma 11.4 says we need $\sqrt{2 \log 2}\sqrt{365} \approx 22.49$ people. In practice the constant $\sqrt{2 \log 2} \approx 1.1774$ is often replaced with 1.

In (c) the birthdays are hash values, so we have $B = 2^m$. Since $\sqrt{2^m} = 2^{m/2}$ we interpret 'hard to find' as 'requires at least $2^{m/2}$ hashes'.

Hash functions in practice. We have already seen one way to make a hash function. Fix a block cipher of length m and a key k. Chop the message x into blocks $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, such that each $x^{(i)} < 2^m$. Let $b^{(i)} \in \mathbb{F}_2^m$ be the binary form of $x^{(i)}$. Then apply the block cipher in cipher block chaining mode (see page 40), to get

$$\begin{aligned} y^{(1)} &= e_k(b^{(1)}) \\ y^{(2)} &= e_k(y^{(1)} + b^{(2)}), \\ &\vdots \\ y^{(t)} &= e_k(y^{(t-1)} + b^{(t)}) \end{aligned}$$

The final ciphertext $y^{(t)} \in \mathbb{F}_2^m$ depends on the entire message x in a complicated way, so is a good choice for the hash value.

Example 11.5 (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 can break (b) when the number of rounds is reduced to 57, and (c) when the number of rounds is reduced to 46.

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 the 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 11.6. 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 11.7. 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[20], by appending $d_a(h(x))$, to get $\big(x, d_a(h(x))\big)$.

[20]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.

Signing the message ensures that only Alice can transfer Alice's bitcoins. But as described so far, Alice can double-spend: a few minutes later she can sign another message $\big(x', d_a(h(x'))\big)$ where x' says 'I Alice give Charlie the bitcoin b'.

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

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

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[21]. 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 Bob receives $\big(x', d_a(h(x'))\big)$, he looks to see if there is are blocks already containing a transaction involving the bitcoin b mentioned in x'. When Bob finds $\big(x, d_a(h(x))\big)$ 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.[22]

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).

[21]At the time of writing 72 zeros are required: see Problem Sheet 9, Question 9 for the implications for the bitcoin economy.

[22]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.

Extra: factoring n given an RSA private key (n, r). (This was a quiz at the end of lecture 22.) Suppose we learn n and the decryption exponent r. We already know the public key (n, c). Hence we can compute cr. By choice of r, we know that $cr \equiv 1 \bmod (p-1)(q-1)$. Hence $cr - 1$ is a multiple of $(p-1)(q-1)$.

Let t be obtained by dividing $cr - 1$ by small odd primes until a factor is found. There is a good chance that either $p - 1$ 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 $\mathrm{hcf}(x^t - 1, n) = p$.

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 11.8. 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]`.[23] 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 $cr - 1 = 162\,954 \ldots 674\,816 \approx 1.6 \times 10^{53}$ and by trial division,

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

in MATHEMATICA, finds that the smallest prime factors of $cr - 1$ are $2, 3, 43, 617, \ldots$. Since 9 divides $cr - 1$, it is possible that 3 divides both $p - 1$ and $q - 1$, so instead he uses 43 and takes $t = (cr-1)/43$. 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]`. The highest common factor turns out to be p.

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.

[23]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.