# 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 (MC201), Friday 4pm (MC336).

**Extra lecture for MSc students doing MT5462:** Friday 9am (MC201).

**Office hours in McCrea 240:** Tuesday 3pm, Wednesday 10am, Thursday 11am or by appointment.

*Date*: First Term 2017/18.

## 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 (one copy on three day 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 at noon on Wednesday 9th October. (If you prefer, hand in at the Monday lecture.) Answers to the preliminary problem sheet will be posted on Moodle on Monday 2nd October. The second Friday lecture will usually be 'flipped', and should give you a good start on the problem sheets.

**Moodle.** All handouts, problem sheets and answers will be posted on Moodle. You should find a link under 'My courses', but if not, go to `moodle.royalholloway.ac.uk/course/view.php?id=380`.

**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.** The 'Bonus question' at the end of each problem sheet, any 'optional' questions, 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.

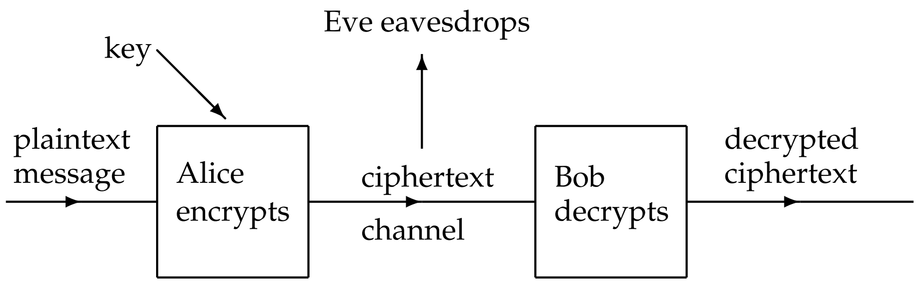
**If you can do the compulsory questions on problem sheets, know the definitions and main results from lectures, and can prove the results whose proofs are marked as examinable in these notes, then you should do very well in the examination.**

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

### 1. Introduction: security requirements

**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, 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 Eve 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.**

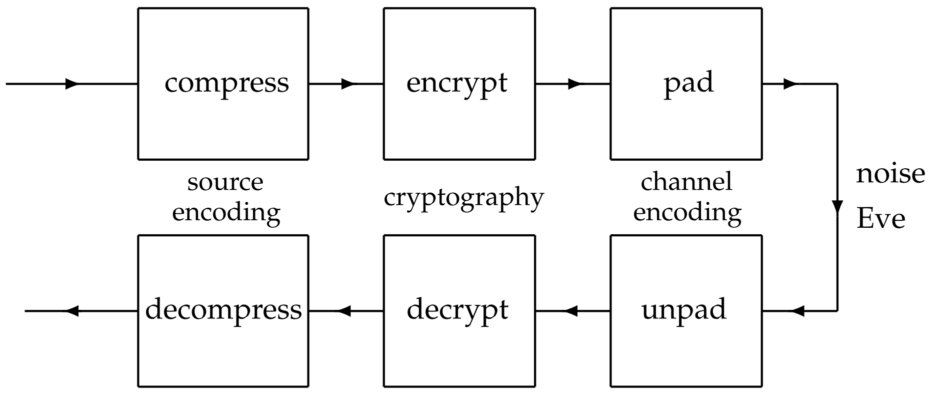
(1) If you encrypt a file using a password on your computer, you care most about confidentiality and data integrity. In this case, you are Alice, and Bob is you a week later. The channel is the hard-disk (or SSD) in your computer.

(2) Using online banking to make a payment, the bank's main security requirements are authentication and non-repudiation. It is now considered 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.

(3) One online chess site requires a password to login, but the moves are then sent unencrypted. Here the main concern is authentication, ensuring that no-one else can steal your account.

Remarkably we will see that Alice and Bob do not need to exchange the key before they communicate. Instead they can establish a shared secret key over an insecure channel, and even authenticate[1] each other, by using Public Key Cryptography. We will learn how in Part D of the course. Our course MT366/466/5466 goes into more detail.

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 to write plaintexts in lower case and ciphertexts in upper case.

*Caesar and substitution ciphers.*

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

**Exercise 2.2.**

(a) Malcolm (the mole) knows that the plaintext 'apple' was encrypted as `CRRNG`. What is the key?

[1]Authentication raises an interesting distinction between your identity (who you are), and your identifiers (username, email address, fingerprint, and so on). For instance, every bitcoin includes, in its blockchain, every transaction it has been part of: this includes the identifiers of all the parties involved. But provided the connection between your identifier and your identity is kept secret, you can still use Bitcoin for anonymous transactions. We will study the SHA-256 hash function used in bitcoin in Part E of the course.

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

(c) Repeat (b) supposing the intercepted ciphertext is `GVTJPO`. Suppose Eve later intercepts `XKIX`. What can she conclude?

| | | | | | |
|---|---|---|---|---|---|
| A | 0 | `ABCDEFGHIJKLMNOPQRSTUVWXYZ` | N | 13 | `NOPQRSTUVWXYZABCDEFGHIJKLM` |
| B | 1 | `BCDEFGHIJKLMNOPQRSTUVWXYZA` | O | 14 | `OPQRSTUVWXYZABCDEFGHIJKLMN` |
| C | 2 | `CDEFGHIJKLMNOPQRSTUVWXYZAB` | P | 15 | `PQRSTUVWXYZABCDEFGHIJKLMNO` |
| D | 3 | `DEFGHIJKLMNOPQRSTUVWXYZABC` | Q | 16 | `QRSTUVWXYZABCDEFGHIJKLMNOP` |
| E | 4 | `EFGHIJKLMNOPQRSTUVWXYZABCD` | R | 17 | `RSTUVWXYZABCDEFGHIJKLMNOPQ` |
| F | 5 | `FGHIJKLMNOPQRSTUVWXYZABCDE` | S | 18 | `STUVWXYZABCDEFGHIJKLMNOPQR` |
| G | 6 | `GHIJKLMNOPQRSTUVWXYZABCDEF` | T | 19 | `TUVWXYZABCDEFGHIJKLMNOPQRS` |
| H | 7 | `HIJKLMNOPQRSTUVWXYZABCDEFG` | U | 20 | `UVWXYZABCDEFGHIJKLMNOPQRST` |
| I | 8 | `IJKLMNOPQRSTUVWXYZABCDEFGH` | V | 21 | `VWXYZABCDEFGHIJKLMNOPQRSTU` |
| J | 9 | `JKLMNOPQRSTUVWXYZABCDEFGHI` | W | 22 | `WXYZABCDEFGHIJKLMNOPQRSTUV` |
| K | 10 | `KLMNOPQRSTUVWXYZABCDEFGHIJ` | X | 23 | `XYZABCDEFGHIJKLMNOPQRSTUVW` |
| L | 11 | `LMNOPQRSTUVWXYZABCDEFGHIJK` | Y | 24 | `YZABCDEFGHIJKLMNOPQRSTUVWX` |
| M | 12 | `MNOPQRSTUVWXYZABCDEFGHIJKL` | Z | 25 | `ZABCDEFGHIJKLMNOPQRSTUVWXY` |

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

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

then $e_\pi(\text{hello there}) = \text{SVOOL GSVIV}$. (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 $\pi$ shifts each letter forward $s$ times.

**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 2.4.** 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 the 13 most common 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 |

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 surely 'a', and 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.

**Exercise 2.5.**

(a) After deciphering, Eve knows that $\pi(a) = U$, $\pi(b) = T$, and so on. Does she know the key $\pi$?

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

The substitution cipher is weak because the same permutation is applied to each letter of the plaintext. Choosing a different permutation for each letter, even if it has to be a Caesar shift, gives a stronger cipher.

**Lecture 3**

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

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

Using this bijection we identify a word of length $\ell$ 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 $x \mapsto x + s \bmod 26$.

**Definition 2.6.** The key $k$ for the *Vigenère cipher* is a word. Suppose that $k$ has length $\ell$. 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 2.7.** Take $k =$ emu, so $k$ has length 3. Under the bijection between letters and numbers, emu $\longleftrightarrow (4, 12, 20)$. The table below shows that

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

| $i$ | 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 |
| $k_i$ | 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 |

**Exercise 2.8.**

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

`QXNURA`, `QMUUFM`, `QNRFLX`.

These come from taking every 2nd, 3rd and 4th position in the ciphertext `QQYXMNQU`... above, starting at the second `Q`, supposing the plaintext continues '...near the tree'.

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

This gives some motivation for the following statistic.

**Definition 2.9.** 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 2.10.** 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 2.11.** *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 2.12.** *Given a Vigenère ciphertext, split it into groups by taking every $\ell$-th letter for all small $\ell$, as in Exercise 2.8. If the ciphertext is long enough, the Index of Coincidence will be greatest at the key length. Each split ciphertext is the output of a Caesar cipher; assuming the most common letter is the encryption of 'e' determines the shift.*

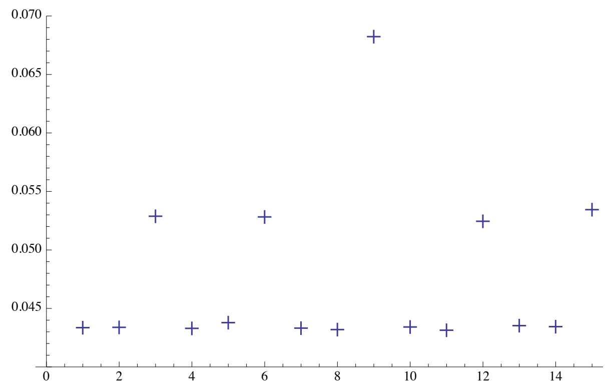
**Example 2.13.** The final 554 words (or 2534 characters) of Chapter 1 of *Persuasion* by Jane Austen begin

> 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, . . . .

Encrypted using the Vigenère cipher with key 'secretkey', the ciphertext is `KYEYAXBICDMBRFXDLCDPKFXLCILLMOVRMCEL` . . . .

The graph overleaf shows the mean Index of Coincidence when the ciphertext is split by taking every $\ell$-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 2.4) 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 2.14.** Explain why there are smaller peaks at 3, 6, 12 and 15 in the plot of Indices of Coincidence above.

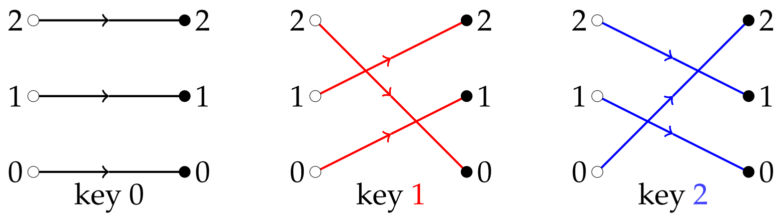
*Statistical methods.* A better way to finish the decryption, which is reliable on smaller ciphertexts, re-uses the Index of Coincidence: see Sheet 1. The $f_i^2$ in the numerator of the formula in Lemma 2.11 may remind you slightly of the $\chi^2$-test: the connection is explored in the optional question 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 recent shocking defeat of the world Go champion by AlphaGo.

## 3. CRYPTOSYSTEMS, ATTACK MODELS AND PERFECT SECRECY

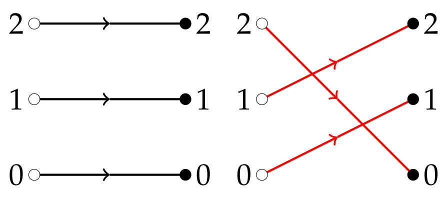
**Lecture 4**

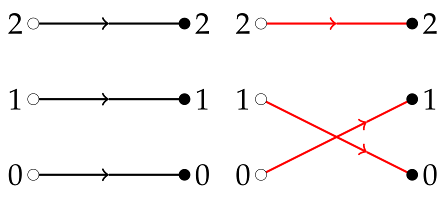
*Cryptosystems.* The three different encryption functions for the Caesar cipher with 'alphabet' $\{0, 1, 2\}$ are shown in the diagram below.

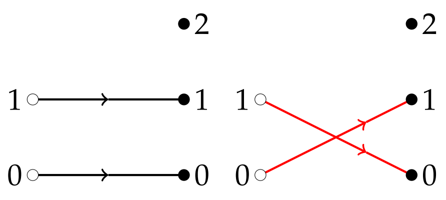


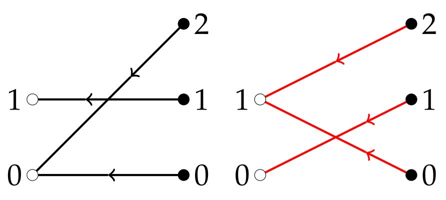
**Exercise 3.1.** Which of the diagrams overleaf show plausible cryptosystems?

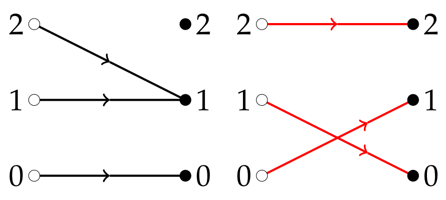
Putative cryptosystems with keys $k$ (black, first) and $k'$ (red, second). Note that in (iv) the decryption functions are shown.

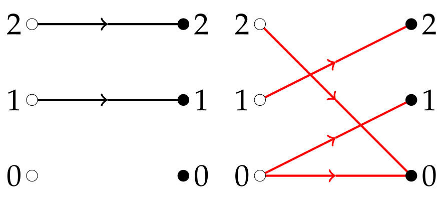
(i)


(ii)


(iii)


(iv)


(v)


(vi)


This gives some motivation for the following definition.

**Definition 3.2.** 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 $k \in K$,

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

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*.

Provided the decryption maps are defined in the unique way so that $(\star)$ holds, the Caesar cipher, substitution cipher and Vigenère cipher and the ciphers shown in Exercise 3.1(i) and (ii) are all cryptosystems. (In each case the encryption functions are bijective: see remark after Exercise 3.4.)

In Exercise 3.1(iii) we have to specify $d_k(2)$ and $d_{k'}(2)$, but any choice gives a cryptosystem with $\mathcal{P} = \{1, 2\}, \mathcal{C} = \{0, 1, 2\}$ and $\mathcal{K} = \{k, k'\}$, such that $e_k(x) = x$ and $e_{k'}(x) = 1 - x$ for all plaintexts $x$. We therefore allow (iii) to define a cryptosystem.

**Exercise 3.3.**

(a) What is special about ciphertext 2 in (iii)?

(b) Define $e_k$ and $e_{k'}$ so that (iv) becomes a cryptosystem. How many choices did you have? Should (iv) be allowed as the definition of a cryptosystem?
(c) What is the problem with (v)?
(d) What are the two problems with (vi)?

**Exercise 3.4.** Prove that the encryption functions in a cryptosystem are injective and that the decryption functions are surjective.

In particular, if $\mathcal{P} = \mathcal{C}$ then, since an injective function between two sets of the same size is bijective, the encryption functions are bijective and uniquely determine the decryption functions.

*Affine cipher.* 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 $7 + 8 \equiv 4$ mod 11 and $7 \times 8 \equiv 1$ mod 11.

**Example 3.5.** 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 maps are defined by $e_{(a,c)}(x) = ax + c \bmod p$. The decryption maps 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 $ab + rp = 1 \bmod p$; then $ab \equiv 1 \bmod p$.

**Lecture 5**

**Exercise 3.6.** 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 Malcolm knows that $e_{(a,c)}(1) = 2$. What does he learn about the key? What happens if he later learns $e_{(a,c)}(0)$?

See Problem Sheet 2 for the definition of the affine cipher on $\mathbb{Z}_n$ for general $n$.

*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 Malcolm) is to determine $k$.

- *Known ciphertext*. Eve knows $e_k(x) \in \mathcal{C}$.
- *Known plaintext and ciphertext*. Malcolm knows $x \in \mathcal{P}$ and $e_k(x) \in \mathcal{C}$.
- *Chosen plaintext*. Malcolm may choose any $x \in \mathcal{P}$ and is given the encryption $y = e_k(x)$.
- *Chosen ciphertext*. Malcolm 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 3.7.**

(1) All the cryptosystems we have seen so far are broken by a chosen plaintext attack. The affine cipher requires two choices and by Question 3 on Sheet 1, the substitution cipher and the Vigenère cipher just one.

(2) 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.

(3) In a known plaintext attack it might be impossible to determine the key because there are two different keys $k, k'$ such that $e_k(x) = e_{k'}(x) = y$. (This is the case when $x = 2$ in Exercise 3.1(ii).) This is rarely a problem in practice.

*Probability and perfect secrecy.* We agreed in Exercise 3.6 that Eve learned nothing about the plaintext by observing ciphertext 2. One way to make this intuitive idea mathematically precise uses probabilities.

Fix a cryptosystem in our usual notation. Suppose that the plaintext $x \in \mathcal{P}$ is sent with probability $p_x$. Let $X$, $Y$ and $K$ be the random variables standing for the plaintext, ciphertext and key, respectively.[2]

**Assumption 3.8.** The plaintext $X$ and the key $K$ are independent.

[2]There are notes on Moodle reviewing basic probability theory. See also the preliminary problem sheet for some practice questions. To be very formal (i.e. you are welcome to stop reading this footnote now), $X$, $Y$ and $K$ are the functions defined on the probability space $\Omega = \mathcal{K} \times \mathcal{P} \times \mathcal{C}$ by $K(k,x,y) = k$, $X(k,x,y) = x$ and $Y(k,x,y) = y$ ; if key $k \in \mathcal{K}$ is used with probability $r_k$ then the probability measure on $\Omega$ is defined by

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

Typically the key is chosen first, and the plaintext does not refer to the key. So Assumption 3.8 holds in practice.

**Example 3.9.** Suppose in each case that the keys are used with equal probability.

(a) In Exercise 3.1(i), $\mathbf{P}[Y=1] = \frac{p_0+p_1}{2}$ and

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

These probabilities are usually not the same as $p_0, p_1, p_2$. (Just take $p_2 \neq 0$.) Hence an Eve intercepting ciphertext 1 learns something about the plaintext.[3]

(b) In the Caesar cipher on $\{0,1,2\}$, shown before Exercise 3.1, we have $\mathbf{P}[X=x|Y=y] = p_x$ for all $x, y \in \{0,1,2\}$. Knowing the ciphertext tells Eve nothing about the plaintext.

**Definition 3.10.** A cryptosystem has *perfect secrecy* if $\mathbf{P}[X=x|Y=y] = p_x$ for all plaintexts $x \in \mathcal{P}$ and all ciphertexts $y \in \mathcal{C}$ such that $\mathbf{P}[Y=y] > 0$.

By Example 3.9(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 get the cryptosystem in Example 3.9(a), which does not have perfect secrecy.

The aim of the remainder of this section is to prove a theorem (originally due to Shannon) describing cryptosystems with perfect secrecy.

**Lemma 3.11.** *A cryptosystem has perfect secrecy if and only if*

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

*for all plaintexts $x \in \mathcal{P}$ such that $p_x > 0$ and all ciphertexts $y \in \mathcal{C}$.*

The second hypothesis of Shannon's theorem requires that each ciphertext may be sent. It excludes cryptosystems such as Exercise 3.1(c) where there is a ciphertext (number 2) that is never used, and some other 'degenerate' cases.

**Lecture 6**

**Theorem 3.12** (Shannon 1949). *Suppose a cryptosystem (in our usual notation) has perfect secrecy and that $\mathbf{P}[Y=y] > 0$ for all $y \in \mathcal{C}$.*

[3] In the language of Bayesian statistics, Eve's posterior probabilities are different to her prior probabilities. Again, if you do not find this remark helpful, please ignore it.

(a) *For each $x \in \mathcal{P}$ such that $p_x > 0$ and each $y \in \mathcal{C}$ there is a key $k$ such that $e_k(x) = y$.*

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

(c) *Suppose $|\mathcal{K}| = |\mathcal{C}|$. For all $x \in \mathcal{P}$ such that $p_x > 0$ and all $y \in \mathcal{C}$ there exists a unique key $k \in \mathcal{K}$ such that $e_k(x) = y$. Moreover for each fixed $y \in \mathcal{C}$, the keys $k$ such that $e_k(x) = y$ for some $x \in \mathcal{P}$ with $p_x > 0$ are used with equal probability.*

**Lecture 7**

**Correction.** As stated above, part (c) is what I proved in lectures. In the lecture I wrote 'Moreover each key is used with equal probability', which I realised later is too vague.

**Corollary 3.13.** *Suppose a cryptosystem (in our usual notation) has perfect secrecy and that*

(i) $\mathbf{P}[Y = y] > 0$ *for all* $y \in \mathcal{C}$;
(ii) $|\mathcal{K}| = |\mathcal{C}|$;
(iii) $p_x > 0$ *for all* $x \in \mathcal{P}$.

*Then for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$ there exists a unique key $k \in \mathcal{K}$ such that $e_k(x) = y$. Moreover each key is used with equal probability.*

*Proof.* This is immediate from (c) in Theorem 3.12. □

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 that the converse of Corollary 3.13 is true.

## 4. Entropy and key uncertainty

*Entropy.* The entropy of a random variable is a measure of how uncertain it is. The right way to make this notion precise was found by Shannon[4].

**Definition 4.1.** The *entropy* $H(X)$ of a random variable $X$ taking values in a finite set $\mathcal{R}$ is

$$H(X) = \sum_{x \in \mathcal{R}} \mathbf{P}[X = x] \log_2 \frac{1}{\mathbf{P}[X = x]}.$$

[4]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.

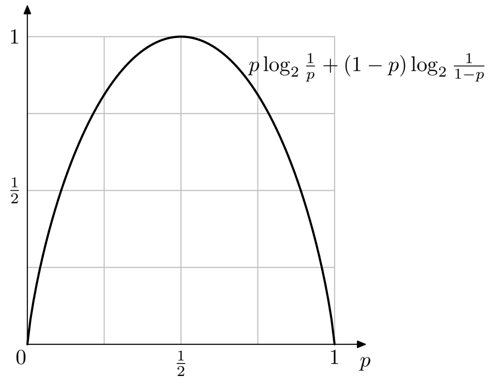
Note that $\log_2$ is the base 2 logarithm, so $\log_2 2^n = n$ for all $n \in \mathbb{N}_0$. If $\mathbf{P}[X = x] = 0$ then $\mathbf{P}[X = x] \log_2 \frac{1}{\mathbf{P}[X=x]}$ should be interpreted as its limiting value of 0. Since $\log_2(1/\mathbf{P}[X = x]) = -\log_2 \mathbf{P}[X = x]$ we have

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

You might prefer to work with this form, but please note the minus sign!

**Example 4.2.**

(1) Suppose $X$ records a single coin flip of a coin, biased to land heads with probability $p$. Then $H(X) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}$, as shown in the graph below.



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.

**Lecture 8**

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

(3) Consider the cryptosystem in Exercise 3.1(iii). Suppose that $\mathbf{P}[X = 0] = p$, and so $\mathbf{P}[X = 1] = 1 - p$, and that the keys are used with equal probability $\frac{1}{2}$. Then

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

*Exercise:* show that $H(Y) = 1$.

(4) Consider the affine cipher on $\mathbb{Z}_5$, as in Exercise 3.6. The keys are all $(a, c)$ with $a \in \{1, 2, 3, 4\}$ and $c \in \{0, 1, 2, 3, 4\}$. If each key is used with equal probability then, by (2), $H(K) = \log_2 20 \approx 4.322$. After Malcolm observes that $e_{(a,c)}(1) = 2$, he knows that

$$(a, c) \in \{(1, 1), (2, 0), (3, 4), (4, 3)\}.$$

Each possibility is equal likely, so the entropy in $K$, given Malcolm's observation, is $\log_2 4 = 2$.

**Exercise 4.3.**

(a) Suppose in Example 4.2(2) that the keyspace is $\{0, 1, \ldots, n-1\}$. If $n = 2^m$, how many yes/no questions does it take to learn the key? If $n = 26$ how many questions do you need?

(b) A friend chooses $0, 1, 2, 3, 4$ with probabilities $\frac{1}{2}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}$, respectively. How many yes/no questions do you need, on average, to guess her number?[5] Would your answer change if the probabilities changed to $\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{2}$?

This motivates the following informal interpretation of entropy:

$$\boxed{\text{A random variable has entropy } h \text{ if and only if you can learn its value by asking, on average, } h \text{ well-chosen yes/no questions.}}$$

For this reason entropy is often regarded as measured in bits.

*Conditional entropy and key uncertainty.*

**Definition 4.4.** Let $K$ and $Y$ be random variables each taking values in finite sets $\mathcal{K}$ and $\mathcal{C}$, respectively. The *joint entropy* of $K$ and $Y$ is defined by

$$H(K, Y) = \sum_{k \in \mathcal{K}} \sum_{y \in \mathcal{C}} \mathbf{P}[K = k \text{ and } Y = y] \log_2 \frac{1}{\mathbf{P}[K = k \text{ and } Y = y]}.$$

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

$$H(K|Y = y) = \sum_{k \in \mathcal{K}} \mathbf{P}[K = k | Y = y] \log_2 \frac{1}{\mathbf{P}[X = k | Y = y]}.$$

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 $\mathcal{K} \times \mathcal{C}$.

In Example 4.2(4) we found $H(K) = \log_2 20$. After observing that $e_{(a,c)}(1) = 2$, there were four possible keys. Correspondingly,

$$H\big(K|(X, Y) = (1, 2)\big) = \log_2 4 = 2.$$

**Example 4.5.** Consider the Caesar cryptosystem in which all 26 keys are equally likely. What is $H(K)$? Suppose, as Eve, you observe the ciphertext `ACCB`. What is $H(K|Y = \texttt{ACCB})$? What if instead you observe `NCYP`?

[5]Probably you started by asking 'is it 0?', and then (if necessary) doing a binary search on the remaining four options. This corresponds to the Fano (and Huffman code) for the probability distribution $\frac{1}{2}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}$, with codewords 0, 100, 101, 110, 111, which you will have seen if you are doing MT362/462/5462.

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'.)

**Lecture 9**

**Lemma 4.6** (Chaining Rule)**.** *Let $K$ and $Y$ be random variables. Then*

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

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

**Lemma 4.7.** *Let $K$ and $X$ be random variables.*

(a) *If $K$ and $X$ are independent then $H(K, X) = H(K) + H(X)$.*
(b) *If $f$ is a bijective function then $H(f(X)) = X$.*

The proof of (a) is Question 1 on Sheet 3. The idea behind (b) is the same as the final part of Exercise 4.3(b). If this does not convince you then please see the optional Question 7 on Sheet 2.

**Theorem 4.8.** *Take a cryptosystem in our usual notation. Then*

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

*Alphabetic ciphers: the one-time pad.* Let $\mathcal{A} = \{a, b, \ldots, z\}$ be the alphabet. We apply Theorem 4.8 to cryptosystems where $\mathcal{P} = \mathcal{C} = \mathcal{A}^n$ for some $n \in \mathbb{N}$. Examples include substitution ciphers (with spaces disallowed) and the Vigenère cipher.

To indicate that plaintexts and ciphertexts have length $n$, we write $X_n$ and $Y_n$ rather than $X$ and $Y$.

**Exercise 4.9.** If $Y_n$ is equally likely to be each element of $\mathcal{A}^n$, what is $H(Y_n)$?

We suppose only those $x \in \mathcal{A}^n$ that make good sense in English have $p_x > 0$. Thus if $n = 8$ then 'abcdefgh' and 'goodwork' are both plaintexts, but $\mathbf{P}[X = \text{'abcdefgh'}] = 0$ whereas $\mathbf{P}[X = \text{'goodwork'}] > 0$.

Shannon estimated[6] that, with this assumption, $H(X_n)$ is at most $1.5n$. This is much less than the $(\log_2 26)n \approx 4.700n$ from Exercise 4.9. (Taking into account the frequency distribution of characters, but no further redundancy, gives $4.195n$, still much too high.)

Thus the per-letter redundancy of English is at least $\log_2 26 - 1.5 \approx 3.200$. Let $R = 3.200$.

Lecture 10

**Example 4.10** (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 maps 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, $x_i + k_i$ is computed by converting $x_i$ and $k_i$ to numbers and adding modulo 26. (In fact the one-time pad is the Vigenère cipher when the key has the same length as the plaintext.) For example, if $n$ was fixed as 8,

$$e_{\text{abcdefgh}}(\text{goodwork}) = \text{gpqgatxr}$$

since

$$\begin{aligned} a &\longleftrightarrow 0, \quad g \longleftrightarrow 6, \quad 0+6 = 6 \longleftrightarrow g, \\ b &\longleftrightarrow 1, \quad o \longleftrightarrow 14, \quad 1+14 = 15 \longleftrightarrow p \end{aligned}$$

and so on. Suppose that all keys are equally likely. Then

$$\begin{aligned} H(X_n) &\approx (\log_2 26 - R)n \\ H(K) &\approx (\log_2 26)n \\ H(Y_n) &\approx (\log_2 26)n \\ H(K|Y_n) &\approx (\log_2 26 - R)n. \end{aligned}$$

Since $H(K|Y_n) = H(X_n)$, the one-time pad is safe against a known ciphertext attack: the key remains just as uncertain as the plaintext

**Exercise 4.11.** In the one-time pad of length $n$, what are $H(K|(X_n, Y_n))$ and $H(X_n|Y_n)$? What does this imply about the security of the one-time pad against a known plaintext/ciphertext attack?

*Unicity distance.* We end by considering cryptosystems, like the Vigenère cipher, where the plaintexts and ciphertexts can be much longer than the key.

**Exercise 4.12.** Show that if $Y_n$ is equally likely to be each element of $\mathcal{A}^n$ then $H(Y_n) - H(X_n) = Rn$ and so

$$(\dagger) \qquad H(K|Y) = H(K) - Rn.$$

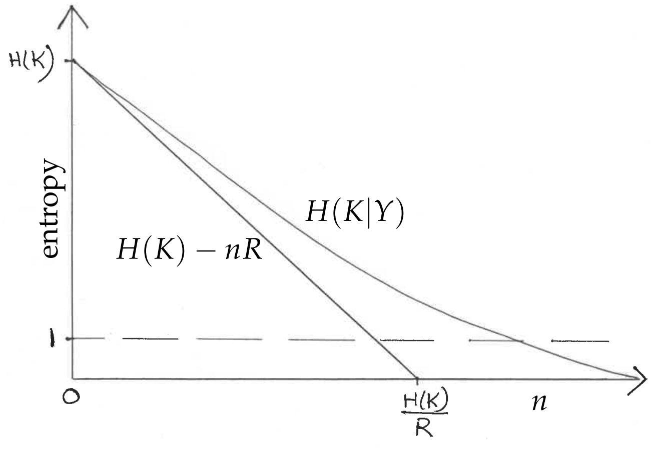
What is the largest $n$ for which (†) could hold with equality?

In his 1949 paper[7], Shannon argues that (†) should be a good approximation to $H(K|Y_n)$, for a large class of cryptosystems, including the Vigenère cipher. The approximation is best when $|\mathcal{K}|$ is large and $n$ is small.

[6]You can play the game Shannon invented to make this estimate online at `https://repl.it/LX00/2`.

[7]*Communication theory of secrecy systems*, Bell Systems Technical Journal (1949) **28**, 656–715

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



In particular, Shannon proved $H(K|Y_n)$ is as shown in the graph above when the cryptosystem is the random cipher, in which the encryption functions and decryption functions are chosen at random, subject to $(\star)$ in Definition 3.2.

**Definition 4.13.** The quantity $H(K)/R$ is the *unicity* distance of the cryptosystem.

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

In practice, the various assumptions in Shannon's argument mean more ciphertext may well be required. But even if the key is not determined completely, this may not be a problem for the attacker.

**Exercise 4.14.** In Question 2 on Sheet 1, the ciphertext $y$, of length 356 (without spaces), determined the key $\pi$ up to $\pi(\mathrm{j}), \pi(\mathrm{x}), \pi(\mathrm{z}) \in \{\mathtt{F}, \mathtt{S}, \mathtt{V}\}$. Assuming equally likely keys, what is $H(K|Y_{356} = y)$?

You are not expected to remember details of the example below. But I hope you will find it interesting! It should give you some flavour of how computing and mathematical arguments come together in cryptography.

**Example 4.15.**

(i) The unicity distance for the substitution cipher is $\log(26!)/R \approx 88.382/3.200 = 27.6$. So 28 characters of ciphertext should, in theory, determine most of the key.

For instance the first 28 characters of the ciphertext in Question 2 on Sheet 1 are (with extra spaces) `XNKWBMOW KWH JKXKRJKRZJ RA KWRJ`. A computer search using a corpus of about 70000 words gives 13 decryptions, all of the form 'although th? statistics i$\star$ this'; where $? \in \{\mathrm{e}, \mathrm{y}\}$ and the only plausible choice for $\star$ is $n$. This essentially unique decryption is in good agreement with Shannon's argument.

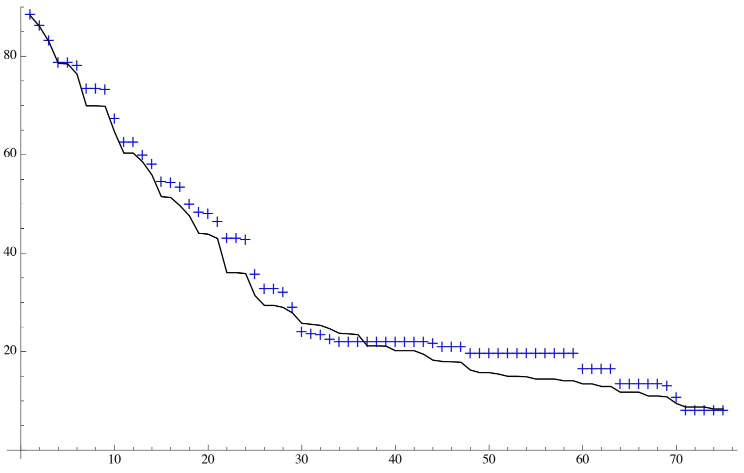
Since 12 characters do not appear in the ciphertext, $H(K|Y = y_{28}) = \log_2 12! = 28.3$. But since $\pi$ is determined on the most common plaintext letters, further decryptions will not be hard.

(ii) Suppose that the plaintext is made by concatenating arbitrary four letter English words in the New General Service List[8]. There are 493 such words, so $H(X_{4m}) = (\log_2 493)m = 8.945m$, compared with $(4\log_2 26)m \approx 18.811m$ for an arbitrary string of $4m$ characters. The per-character redundancy is

$$\frac{4\log_2 26 - \log_2 493}{4} \approx 2.464$$

and so Shannon's argument says that the unicity distance for the substitution cipher is about $\log_2(26!)/2.464 = 35.868$. Therefore about 9 words should determine a large part of the key.

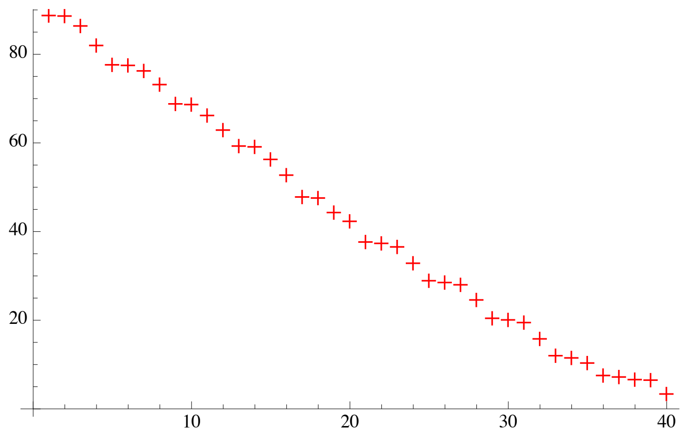
The blue points in the graph below show $H(K|Y_n)$ for the randomly chosen plaintext (shown with spaces for readability) 'case sale thin coal bore will much fuel gain soil site wear form fill wise task bend wild pray easy'. The black line shows the average of over 600 randomly chosen plaintexts.



Again there is good agreement with Shannon's argument. In particular, the slope (the amount we learn per new character) drops off noticeably at about the unicity distance.

[8]This is a basic vocabulary of 2500 words aimed at English learners, available online at `http://www.newgeneralservicelist.org`.

The final graph overleaf shows $H(K|Y_n)$ for the contrived plaintext 'away bank city drug exam from have lazy joke pose' chosen to contain every letter except 'q'.



Almost all the key is known by the unicity distance. In fact, by the final character, there are just 6 decryptions consistent with the NGSL; one is the plaintext, another is 'away bank city drug exam from save lazy joke hope', obtained by permuting the plaintext by the 3-cycle $h \mapsto s \mapsto p \mapsto h$.

**Exercise 4.16.**

(a) Why is it unnecessary to specify the ciphertexts in Example 4.15(2)?
(b) Explain why the biggest drops in the two graphs in this example are every 4th character.

*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'.

More precisely, the attacker is assumed to know the set of plaintexts $\mathcal{P}$, the set of ciphertexts $\mathcal{C}$ and the keyspace $\mathcal{K}$. Moreover, for each key $k \in \mathcal{K}$, she knows the encryption function $e_k$ and decryption function $d_k$. But she does not know which key is being used.[9]

For more (non-examinable) on the advantages and disadvantages of keeping other parts of the cryptosystem secret, try searching for 'security by obscurity' on the web.

[9]In Public Key Cryptography the attacker even knows the function $e_k$ *for the chosen key* $k$. These functions are chosen to be hard to invert, so knowing $e_k$ does not imply knowing $d_k$.

**(B) Stream ciphers**

## 5. Linear Feedback Shift Registers

Kerckhoff's Principle states that 'all the security is in the key'. We saw in Corollary 3.13 that (under reasonable conditions on the possible messages) perfect secrecy requires that each key is used with the same probability. So we need keys that 'look random'. **Lecture 11**

Computers are deterministic: given the same inputs, you always get the same answer. In this section we will see how to get keys that 'look random' out of deterministic algorithms. We will also see some ways to define randomness more precisely.

*Definition of LFSRs.* 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, $\mathbb{F}_2^\ell$ is the set of $\ell$-tuples $(x_0, x_1, \ldots, x_{\ell-1})$ where each $x_i$ is a bit. For brevity we may write this tuple as $x_0x_1 \ldots x_{\ell-1}$. As seen here, we number positions from 0 up to $\ell - 1$.

For example $x = 0111 \in \mathbb{F}_2^4$ is the tuple $(0,1,1,1)$ with entries $x_0 = 0$, $x_1 = 1$, $x_2 = 1$, $x_3 = 1$.

**Example 5.1.** Consider the function $F : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ 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) Starting with $x = 0001$, the sequence $x, F(x), F^2(x), F^3(x), \ldots$ is $(0001, 0010, 0100, 1001, 0011, 0110, 1101, 1010, 0101, 1011, 0111, 1111, 1110, 1100, 1000, 0001, \ldots)$. We stop when we see 0001 again since we then know what happens when we apply $F$.

*Exercise:* What is the set of $m \in \mathbb{Z}$ such that $F^m(x) = x$? Would your answer change if $x$ was replaced with another $x' \in \mathbb{F}_2^\ell$?

**Exercise 5.2.** Repeat Example 5.1 for the function $G : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ defined by $G\big((x_0, x_1, x_2, x_3)\big) = (x_1, x_2, x_3, x_0 + x_1 + x_2 + x_3)$.

The functions $F$ and $G$ are instances of the following definition.

**Definition 5.3.** A *linear feedback shift register* of *width* $\ell \in \mathbb{N}$ 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

Lecture 12

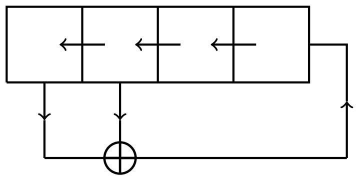
$$F\big((x_0, x_1, \ldots, x_{\ell-2}, x_{\ell-1})\big) = (x_1, \ldots, x_{\ell-1}, \sum_{t \in T} x_t).$$

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*.

We abbreviate 'linear feedback shift register' to LFSR. 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 $\ell - 1$.

**Exercise 5.4.** What is 'linear' about an LFSR?

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



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

The LFSRs $F$ and $G$ seen earlier have width 4 and taps $\{0, 1\}$ and $\{0, 1, 2, 3\}$ respectively.

| | width | taps | feedback function |
|---|---|---|---|
| $F$ | 4 | $\{0, 1\}$ | $f(x_0, x_1, x_2, x_3) = x_0 + x_1$ |
| $G$ | 4 | $\{0, 1, 2, 3\}$ | $g(x_0, x_1, x_2, x_3) = x_0 + x_1 + x_2 + x_3$ |

*The cryptosystem defined by an LFSR.*

**Exercise 5.5.**

(a) Let $F$ be as in Example 5.1. Find the sequence $F^t(0111)_0$ for $t \in \mathbb{N}_0$. (Note that $F^0$ is, by definition, the identity function.)

(b) Let $F$ be an LFSR of width $\ell$ and let $k \in \mathbb{F}_\ell$. Show that $F^t(k)_0 = k_t$ if $0 \leq t < \ell$ and that $F^\ell(k)_0 = f(k_0, \ldots, k_{\ell-1})$.

**Definition 5.6.** Let $F$ be an LFSR of width $\ell$.

Lecture 13

(a) The *keystream* defined by $F$ for key $k \in \mathbb{F}_2^\ell$ is the sequence

$$(k_0, k_1, k_2, \ldots, k_t, \ldots)$$

where $k_t = F^t(k)_0$ for each $t \in \mathbb{N}_0$.

(b) Fix $n \in \mathbb{N}$. 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

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

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

By Exercise 5.5, $k_t = F^t(k)_0$ if $t < \ell$, so, as the notation requires, the keystream starts with $k$. More generally, by Question 2 on Sheet 4,

$$F^s(k) = (k_s, k_{s+1}, \dots, k_{s+\ell-1})$$

for each $s \in \mathbb{N}$. So you can read off $F^s(k)$ from the keystream simply by taking the $\ell$ positions starting with $k_s$ in position $s$.

**Example 5.7.** In Exercise 5.5(a) we found that the keystream for the LFSR $F$ in Example 5.1 with key $k = 0111$ was

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

For example,

$$\begin{aligned} F^3(0111) &= (k_3, k_4, k_5, k_6) = (1,1,0,0) \\ F^{14}(0111) &= (k_{14}, k_{15}, k_{16}, k_{17}) = (1,0,1,1) \\ F^{15}(0111) &= (k_{15}, k_{16}, k_{17}, k_{18}) = (0,1,1,1) = k. \end{aligned}$$

The keystream has period 15. Correspondingly 15 is the smallest number $m$ such that $F^m\big((0,1,1,1)\big) = (0,1,1,1)$.

**Definition 5.8.** We define the *period* of an invertible LFSR $F$ to be the least $m$ such that $F^m = \mathrm{id}$, the identity function.

For example, the LFSRs $F$ and $G$ in Example 5.1 and Exercise 5.2 have periods 15 and 5, respectively.

For cryptographic use we want the period to be large. Small periods are dangerous because the repetition in the keystream means they can be attacked like the Vigenère cipher.

*Cycles and the matrix representation of an LFSR.* To understand how to find LFSRs with long periods we use methods from linear algebra.

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

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

*where*

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

This matrix is invertible if and only if $0 \in T$. So, as conjectured in Lecture 12, an LFSR is invertible if and only if 0 is one of its taps. (You are asked to prove this directly, in a way that gives a stronger result, in Question 4 on Sheet 4.)

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 $\ell$. 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), 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 $\ell$ 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 $M$.*

**Lecture 14**

Motivated by the lemma we define the *minimal polynomial* of an LFSR $F$ of width $\ell$ with taps $T$ to be $g_F(X) = X^\ell + \sum_{t\in T} X^t$.[10]

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 $f(X) \in \mathbb{F}_2[X]$ is a polynomial such that $f(M) = 0$ then $f(X)$ is divisible by the minimal polynomial of $M$.[11]

**Lecture 15**

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

[10]The term 'characteristic polynomial' is also used: this is justified because $g_F$ is the characteristic polynomial of the matrix $M$ representing $F$. This follows from Lemma 5.10, or can be proved directly; see Question 9 on Sheet 4.

[11]*Proof:* let $g(X)$ be the minimal polynomial of $M$. By polynomial division we have $f(X) = q(X)g(X) + r(X)$ where either $r(X) = 0$ or $\deg r(X) < \deg g(X)$. Then $0 = f(M) = q(M)g(M) + r(M) = q(M)0 + r(M) = r(M)$ 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 $f(X)$.

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 7 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 $\ell$ is at most $2^\ell - 1$.

To illustrate Corollary 5.11 we find an LFSR of width 11 with period $2^{11} - 1 = 2047$. One reason why LFSRs are useful is that the period is typically far more than the width.

**Example 5.12.** The MATHEMATICA command

```
    Factor[X^(2^11 - 1) + 1, Modulus -> 2]
```

returns $(1+X)(1+X^2+X^{11})(1+X+X^2+X^4+X^{11})\ldots$. (Here $\ldots$ stands for 185 omitted factors all of degree 11.) The LFSR of width 11 with minimal polynomial $1+X^2+X^{11}$ has taps $\{0,2\}$. By Corollary 5.11, its period is the least $m$ such that $1+X^2+X^{11}$ divides $X^m+1$. The output of

```
    Select[Table[{m,
       PolynomialRemainder[X^m + 1, X^11 + X^2 + 1,
         X, Modulus -> 2]}, {m, 1, 2047}], #[[2]] == 0 &]
```

shows that the least such $m$ is $2^{11} - 1$.

The computation can be much reduced from a lot to nothing using some finite field theory, but this is beyond the scope of this course.[12]

Using the field theory in this footnote it follows that for every $\ell \in \mathbb{N}$ there exists an LFSR of width $\ell$ of maximum possible period $2^\ell - 1$.

*Attacks on LFSRs.* Take an LFSR cryptosystem in which $\mathcal{P} = \mathcal{C} = 2^n$. Suppose, as in the known plaintext/ciphertext attack, we know that $e_k(x) = y$ where $x, y \in \mathbb{F}_2^n$. Then, as seen for the one-time pad, $(k_0, k_1, \ldots, k_{n-1}) = x + y$. We therefore learn the first $n$ bits of the keystream.

Provided $n \geq \ell$ we now know the key. Following Kerckhoff's Principle that 'all the security is in the key', the width $\ell$ of the LFSR and the taps $T$ can be supposed to be known. So we can learn the entire keystream and can decrypt any further messages sent with this key.

[12]Let $\mathbb{F}_{2^\ell}$ be the finite field of size $2^\ell$. The multiplicative group of $\mathbb{F}_{2^\ell}$ is cyclic; let $\alpha \in \mathbb{F}_{2^\ell}$ be a generator of order $2^\ell - 1$. Let $g(X)$ be the minimal polynomial of $\alpha$. Such minimal polynomials are said to be *primitive*. Then $g(X)$ is irreducible of degree $\ell$ and divides $X^m - 1$ if and only if $m$ is a multiple of $2^\ell - 1$. Hence the LFSR with minimal polynomial $g(X)$ has period $2^\ell - 1$. If $\ell$ is prime then any irreducible polynomial of degree $\ell$ is primitive; correspondingly $\mathbb{F}_{2^\ell}$ has no proper subfields except for $\mathbb{F}_2$. For example, $X^7 + 1 = (X+1)(X^3+X+1)(X^3+X^2+1)$ where both cubics are primitive. Thus *any* irreducible polynomial of degree 11 gives an LFSR of maximal possible period $2^{11} - 1$. Example 5.1 and Exercise 5.2 show two of the LFSRs coming from the factorization $X^{15} + 1 = (X+1)(X^2+X+1)(X^4+X^3+X^2+X^1+1)(X^4+X+1)(X^4+X^3+1)$; here the final two factors are primitive, corresponding to the $\phi(15)$ elements of $\mathbb{F}_{2^4}$ of order 15.

In the following example we suppose $\ell$ is known and use linear algebra to determine the taps $T$. In practice the method works even if we have to guess $\ell$: see Question 2 on Sheet 5.

**Example 5.13.** Malcolm the mole knows the plaintext/ciphertext pair

$$\begin{array}{rl} x = & (0,0,1,1,0,1,0,1,1,0,1,0,1,1,0,0,1,1,1,1) \\ y = & (0,0,1,1,1,0,1,0,0,0,0,0,0,1,0,1,0,1,1,1) \\ & \;\;0\;1\;2\;3\;4\;5\;6\;7\;8\;9\;0\;1\;2\;3\;4\;5\;6\;7\;8\;9 \end{array}$$

for an LFSR cryptosystem of width 5, and deduces the keystream starts

$$\begin{array}{rl} x + y = & (0,0,0,0,1,1,1,1,1,0,1,0,1,0,0,1,1,0,0,0) \\ & \;k_0\,k_1\,k_2\,k_3\,k_4\,k_5\,k_6\,k_7\,k_8\,k_9 k_{10} \quad\quad \ldots \quad\quad k_{19} \end{array}$$

The keystream does not obviously repeat, so he guesses that the period of the LFSR is more than 20. Taking the first five bits, Malcolm learns that $k = (0,0,0,0,1)$. By Question 2 on Sheet 4 (see also Example 5.7), he knows that

$$\begin{aligned} F(k) &= (k_1, \ldots, k_5) = (0,0,0,1,1) \\ F^2(k) &= (k_2, \ldots, k_6) = (0,0,1,1,1) \\ F^3(k) &= (k_3, \ldots, k_7) = (0,1,1,1,1) \end{aligned}$$

and so on. The six vectors $k, F(k), \ldots, F^5(k)$ lie in the 5-dimensional vector space $\mathbb{F}_2^5$ so are linearly dependent. By row-reducing the matrix

$$\begin{array}{r} k \\ F(k) \\ F^2(k) \\ F^3(k) \\ F^4(k) \\ F^5(k) \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

or by inspection, he sees that $k + F^4(k) + F^5(k) = (0,0,0,0,0)$. This suggests (in fact proves, using the optional Question 7 on Sheet 4) that the minimal polynomial of the LFSR is $1 + X^4 + X^5$, and so the taps are $\{0,4\}$: this can be checked by computing $F(k)$, $F^2(k)$, ... assuming these taps.

The Berlekamp–Massey algorithm is a faster way to determine the taps that does not require a guess of the width. It will be seen in the **M.Sc.** course. (It is not part of the 362/462 course.)

Given a known ciphertext we can try to guess part of the plaintext, and use the known plaintext/ciphertext attack above. If the ciphertext is so long that the key repeats then the methods used to break the Vigenère cipher and the re-used one-time pad are also applicable.

## 6. Pseudo-random number generation

Lecture 16

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.

We saw after Corollary 5.11 that the maximum possible period of an LFSR of width $\ell$ is $2^\ell - 1$. Given such an LFSR and 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 $k$.

**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 (and later examples). 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 Question 2 of Sheet 4,

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

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

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

In Example 5.5(a) we used this LFSR with the alternative key $(0, 1, 1, 1)$.

(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 $(0, 0, 1)$, $(0, 0, 0)$, $(0, 1)$ and $(0, 0)$. Explain the pattern.

Question 1(c) on Sheet 4 needs the same idea as (c). There was nothing special about $(0, 1, 1, 1)$ here, except that it is non-zero, so any non-zero $x \in \mathbb{F}_2^4$ has a keystream of period 15.

**Proposition 6.2.** *Let $F$ be an invertible LFSR of width $\ell$ 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* $(k_t, \ldots, k_{t+m-1}) = (0, 0, \ldots, 0) \in \mathbb{F}_2^m$.

**Lecture 17**

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

**Exercise 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 the exercise, 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 $\mathrm{Bin}(\frac{1}{2}, n)$ distribution.

(b) Show that the $\chi^2$ 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? (This is the 'monobit test'.) [*Hint:* if $Z \sim N(0, 1)$ then $\mathbf{P}[Z^2 \geq 3.841] \approx 0.05$ and $\mathbf{P}[Z^2 \geq 6.635] \approx 0.01$.]

**Lecture 18**

(d) The 2001 version of FIPS 140-2 required that $9725 < M_0 < 10275$ when $n = 20000$. This requirement was withdrawn[13] in 2002. Suggest a possible reason for this change.

See Question 3 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.

**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

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

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

[13] See the struck-out text on page 57 of `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf`. FIPS stands for Federal Information Processing Standard.

**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. Why is there no need to specify 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 $\ell$. 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 the keystream of a full-width LFSR has a strong randomness property.

## 7. Non-linear stream ciphers

Mathematically an LFSR of width $\ell$ is a function $F : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell$. The domain $\mathbb{F}_2^\ell$ corresponds to the $\ell$ bits stored in the registers: we call these bits the *internal state* of the LFSR. It is updated by the linear function $F$. **Lecture 19**

The cryptosystem in Definition 5.6(b) is trivially broken by a known plaintext/ciphertext attack (see bottom page 27) because every bit of internal state appears, unmodified, in the keystream.

**Example 7.1.** Totally Trusted Transmission Technologies thinks that taking the sum of the keystreams for two LFSRs with different keys should obscure the keys and give a cryptographically strong sequence.

- Let $F$ be the LFSR of width 3 with taps $\{0, 1\}$.
- Let $F'$ be the LFSR of width 4 with taps $\{0, 3\}$.

The periods of $F$ and $F'$ are 7 and 15, maximum possible for their widths.

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

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

TTTT are soon informed by an irate customer that $(u_0, u_1, u_2, \ldots)$ is generated by $F'$. *Exercise:* check this and explain why.

*Exercise:* can the keys $k$ and $k'$ be recovered from $(u_0, u_1, \ldots, u_{19})$? If so, explain how; if not, will this deter attackers?

(b) TTTT decides their error was to use the same LFSR twice. The first 20 bits in the keystreams for $F$ and $F'$ with keys $k = (0,0,1)$ and $k' = (1,0,0,0)$ and their sum $(u_0, u_1, \ldots, u_{19})$ are:

| | |
|---|---|
| $k_i$ | 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 |
| $u_i$ | 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 |

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

The linear algebra method from Example 5.13 or Question 2 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 $\{0, 1, 5, 6\}$.

*Exercise:* check this holds for the first 20 bits.

The optional Question 4 on Sheet 6 shows this holds for all the bits of the sequence, *regardless* of the choice of keys $k$ and $k'$ and that any 7 consecutive bits from $(u_0, u_1, u_2, \ldots)$ determine the keys $k$ and $k'$. So again $(u_0, u_1, u_2, \ldots)$ is weak.

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

**Example 7.2.** A *Geffe generator* is constructed using three LFSRs $F$, $F'$ and $G$ of widths $\ell, \ell'$ 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

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

For example, if $F$ and $F'$ and their keystreams are as in Example 7.1 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

| | |
|---|---|
| $k_i$ | 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 |

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

**Lecture 20**

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 2 on Sheet 6). But it is vulnerable to a correlation attack.

*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 $s \in \mathbb{N}_0$.[14]

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 7.3.** *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 7.3 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. (There are also faster ways to finish: see Question 2(b) on Sheet 6.)

The Geffe cipher is weak because each keystream bit $xy$ is a product biased to 0. This exercise suggests one way to reduce, but not eliminate, this bias.

**Exercise 7.4.** Let $x, y, z$ be independent unbiased bits. Find the correlation between $xy + z$ and $x$, and between $xy + z$ and $z$. **Lecture 21**

We use $xy + z$ as the source of feedback function in the next cipher. It serves as an example of several important ideas, but you are not expected to remember details of how it is defined or attacked.

**Example 7.5.** Let $F$ be the LFSR of width 5 with taps $\{0\}$. The keystream of $F$ with key $(k_0, k_1, k_2, k_3, k_4)$ is simply $(k_0, k_1, k_2, k_3, k_4, k_0, k_1, \ldots)$. The diagram overleaf, as in Exercise 5.2 or Question 3(c) on Sheet 4, shows the cycles of $F$.

Define $Q(x_0, x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4, x_0 + x_1x_2)$.

*Exercise.* Prove that $Q$ is invertible.

[14]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 informal notation should be clear and will be used for the rest of this section.

$$\begin{tikzcd}
10000 \arrow[dd] & 00100 \arrow[l] & \\
& & 01000 \arrow[ul] \\
00001 \arrow[r] & 00010 \arrow[ur] &
\end{tikzcd}
\quad
\begin{tikzcd}
10001 \arrow[dd] & 11000 \arrow[l] & \\
& & 01100 \arrow[ul] \\
00011 \arrow[r] & 00110 \arrow[ur] &
\end{tikzcd}
\quad
\begin{tikzcd}
11001 \arrow[dd] & 11100 \arrow[l] & \\
& & 01110 \arrow[ul] \\
10011 \arrow[r] & 00111 \arrow[ur] &
\end{tikzcd}$$

$$\begin{tikzcd}
00101 \arrow[dd] & 10100 \arrow[l] & \\
& & 01001 \arrow[ul] \\
01010 \arrow[r] & 10100 \arrow[ur] &
\end{tikzcd}
\quad
\begin{tikzcd}
10101 \arrow[dd] & 11010 \arrow[l] & \\
& & 01101 \arrow[ul] \\
01011 \arrow[r] & 01011 \arrow[ur] &
\end{tikzcd}
\quad
\begin{tikzcd}
11011 \arrow[dd] & 11101 \arrow[l] & \\
& & 11110 \arrow[ul] \\
10111 \arrow[r] & 01111 \arrow[ur] &
\end{tikzcd}$$

$$00000 \qquad 11111$$

When $x_1 x_2 \neq 0$, we have $Q(x) = F(x)$. On the remaining $\frac{1}{4}$ of the inputs, shown in red in the diagram, $Q(x)$ and $F(x)$ differ by $(0,0,0,1)$. This means some of the cycles of $F$ are joined up to make $Q$.

*Exercise:* show that $Q$ has cycles of lengths 1, 5, 5, each equal to a cycle of $F$, and one cycle of length 21, obtained by joining up the remaining five cycles of $F$.

Define the *$Q$-state stream* for key $(k_0, k_1, k_2, k_3, k_4)$ by

$$q_s = Q^s(k_0, k_1, k_2, k_3, k_4)_0 \quad \text{for } s \in \mathbb{N}_0$$

For example, since 00011 is in the cycle 00011 $\to$ 00110 $\to$ 01100 $\to$ 11001 $\to \ldots \to$ 11100 $\to$ 110000 $\to$ 10001 $\to$ 00011 of $Q$ of length 21, its $Q$-state stream is

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

of period 21.

Since $Q$ is invertible, any 5 consecutive bits in the $Q$-state stream determine the internal state and hence the key. For example, if $(q_7, q_8, q_9, q_{10}, q_{11}) = (1,1,0,1,1)$ then working back through the state stream above shows that the key is $(0,1,1,1,0)$.

We avoid this weakness by taking the bits in even-numbered positions in the state stream to define the *$Q$-keystream* $u_0, u_1, u_2, \ldots$. For example, the $Q$-keystreams for keys $(0,0,0,1,1)$ and $(1,1,1,0,1)$ are

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

where the bottom row shows the positions in the state stream. [**Correction:** use $u_s$ not $k_s$ for $Q$-keystream, since $u_0, u_1, u_2, \ldots = k_0, k_2, k_4, \ldots$.]

**Exercise 7.6.**

Lecture 22

(a) Check the $Q$-keystream for $(1,1,1,0,1)$ is as claimed. [*Hint:* you can use the $Q$-state stream for $(0,0,0,1,1)$.]

(b) Why is the period of both keystreams 21?
(c) Show by example that 5 consecutive bits in the $Q$-keystream do not, in general, determine the key.

There is an obvious way to generalize $Q$ to arbitrary widths.

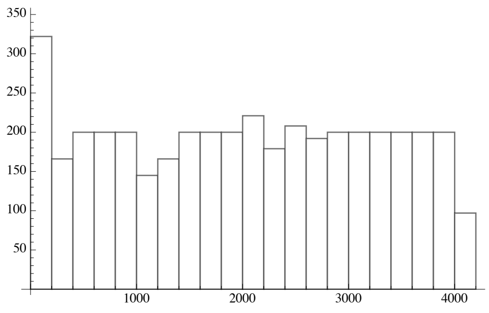
**Exercise 7.7.** Suppose we take $\ell = 12$ and, since the first few bits in the $Q$-keystream are noticeably less random, drop the first 200 bits. For $200 \leq s \leq 1000$, the maximum correlation between a bit $q_s$ of the $Q$-state stream and one of the bits $k_j$ of the key is $\frac{1012}{2^{12}} = \frac{253}{2^{10}} \approx \frac{1}{4}$; with equality when $(s, j) \in \{(751, 0), (752, 1), \dots, (762, 11)\}$. Note that $q_{752} = u_{376}$, and so on, up to $q_{762} = u_{381}$. **[Correction: use $u_s$, not $k_s$ for $Q$-keystream.]**

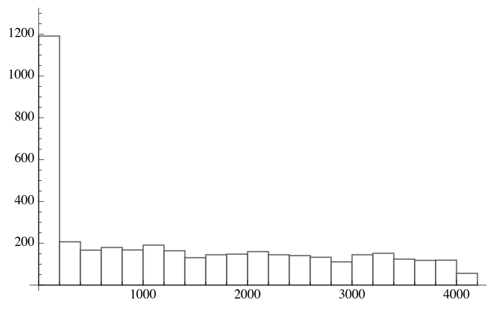
You know $(u_{376}, u_{377}, u_{377}, u_{378}, u_{379}, u_{380}, u_{381})$ and have to guess the key. How should the search be organized?

A naive search going through all $2^{12}$ keys in lexicographic order requires on average 2018.8 guesses. (The code is online at `https://repl.it/NE32/3`.) Ordering the keys so that the 64 keys of the form

$$(\star, u_{376}, \star, u_{377}, \star, u_{378}, \star, u_{379}, \star, u_{380}, \star, u_{381})$$

are tried first, then the $64 \times 6$ keys differing in a single odd numbered position, and so on, reduces the mean number of guesses to 1425.0. The histograms below show the distribution of the number of guesses for the naive search (left) versus the organized search (right).





The distribution is not uniform for the naive search because some 'weak keys' generate the zero keystream, and so are (wrongly, but very quickly) identified as the zero key.

**Remark 7.8.** This improvement can be predicted theoretically. The correlation between $q_{752}$ and $k_1$ is $\mathbf{P}[q_{752} = k_1] - \mathbf{P}[q_{752} \neq k_1]$, or equivalently, $2\mathbf{P}[q_{752} = k_1] - 1$. Therefore $\mathbf{P}[q_{752} = k_1] \approx \frac{1}{2}(1 + \frac{1}{4}) \approx \frac{5}{8}$, and similarly, for $\mathbf{P}[q_{754} = k_3]$, and so on. Therefore the entropy in the key is

$$6 \times f(\tfrac{1}{2}) + 6 \times f(\tfrac{5}{8}) \approx 6 \times 1 + 6 \times 0.9544 = 11.7266$$

where $f(p) = -p \log_2 p - (1 - p) \log_2 p$, as in Example 4.2(1), gives the entropy for each bit. Since on average we find the key halfway through the search, this predicts that $2^{11.7266-1} \approx 1694.45$ guesses (or 'questions about the key') will be required, versus $2^{12-1} = 2048$ for a naive search. In practice the attack is better than this argument predicts.

An attack such as Attack 7.3 or the correlation attack on the $Q$-cipher is said to be *sub-exhaustive* because it finds the key using fewer guesses than brute-force exhaustive search through the keyspace.

**Lecture 23** We end by looking at a modern stream cipher which, like the $Q$-cipher, is defined by mixing addition and multiplication, and like the Geffe cipher, uses multiple shift registers. This combination gives a cipher with no known sub-exhaustive attacks.

**Example 7.9** (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

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

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.[15] The first 1152 bits of the keystream are unusually biased, and so are discarded.

[15]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 24

In stream ciphers a binary plaintext of arbitrary length $n$ is encrypted by adding the first $n$ bits of the keystream for the chosen key. In a block cipher of *block size* $n$, we also have $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^n$, and $\mathcal{K} = \mathbb{F}_2^\ell$, for some $\ell \in \mathbb{N}$, called the *key length*. But typically the plaintext is mixed up with the key in more complicated ways.

Since $\mathcal{P} = \mathcal{C}$, each encryption function $e_k$ for $k \in \mathcal{K}$ is bijective, and the cryptoscheme is determined by the encryption functions.

**Example 8.1.** The binary one-time pad of length $n$ is the block cipher of block size $n$ and keyspace $\mathbb{F}_2^n$ 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 $(x, e_k(x))$ 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 $\mathbb{F}_2^\ell$.

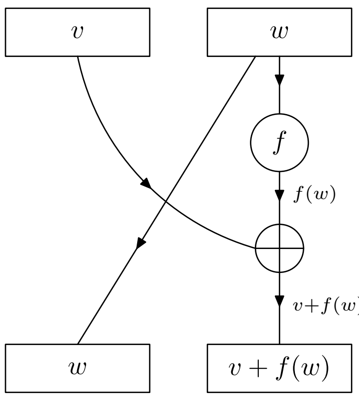
*Feistel networks.*

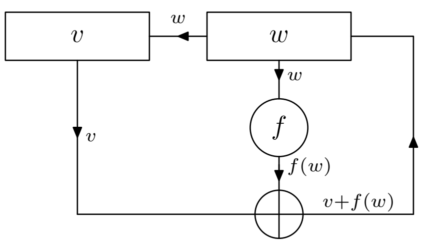
**Definition 8.2.** Let $m \in \mathbb{N}$ and let $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$ be a function. Given $v$, $w \in \mathbb{F}_2^m$, let $(v, w)$ denote $(v_0, \ldots, v_{m-1}, w_0, \ldots, w_{m-1}) \in \mathbb{F}_2^{2m}$. The *Feistel network* for $f$ is the function $F : \mathbb{F}_2^{2m} \to \mathbb{F}_2^{2m}$ defined by

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

This can be compared with an LFSR: we shift left by $m$ positions to move $w$ to the start. 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 8.3.** 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 8.4** ($Q$-block cipher)**.** Take $m = 4$ and let

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

(This is two iterations of the $Q$-cipher from Example 7.5, defined with width 4.) 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 8.5.**

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

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

(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 $v^{(2)}$?

**Lecture 25** *Data Encryption Standard (DES).* 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 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

**Exercise 8.6.** 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: 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).* AES is the winner of an open competition to design a successor to DES. It has block size 128 and keyspace $\mathbb{F}_2^{128}$. 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 $\mathbb{F}_{2^8}$. **Lecture 26**

**Example 8.7.** Let $\alpha$ 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 $\alpha$, but whenever you see a power $\alpha^d$ 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

$$\begin{aligned}\alpha^9 &= \alpha \times \alpha^8 \\ &= \alpha(1 + \alpha + \alpha^3 + \alpha^4) \\ &= \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5.\end{aligned}$$

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

**Definition 8.8.** Let $\mathbb{F}_{2^8}$ be the finite field of size $2^8$ as in Example 8.7. 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 8.7, 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 $\mathbb{F}_2^8$.

[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 $\alpha$ 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 Question 5 on Problem Sheet 7. This gives confusion.
- Diffusion comes from a row permutation of the 16 subblocks, organized into a $4 \times 4$ grid:

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

  The columns are then mixed by a linear map, giving further diffusion.
- The round key in $\mathbb{F}_2^{128}$ is added after these two steps.

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 $x^{(1)}, \ldots, x^{(m)} \in \mathbb{F}_2^n$:

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

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

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

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

- 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 2.4 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 and was considered when designing the DES $S$-boxes. 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 $k_{\mathrm{otp}}$.

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

**Attack 9.1.** *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 $\ell$. 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*

**Lecture 27**

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

*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*

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

*Exercise:* Show that $k_{\mathrm{guess}} \in \mathcal{K}_z$ if and only if $k_{\mathrm{guess}} + \Gamma \in \mathcal{K}_z$. Hence there are always evenly many possible keys.

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

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 $k_{\mathrm{otp}}$.

*Attack on the AES S-box.* We apply Attack 9.1 to a cryptosystem based on the pseudo-inverse function $P : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ used in AES.

**Example 9.2.** Let $n = 8$, $\ell = 8$. 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 9.1 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 $\Delta$ then $k_{\text{guess}} \in \mathcal{K}_z$ and $k_{\text{guess}}$ is either $k$ or a false key. Otherwise we reject $k_{\text{guess}}$.

By Exercise 9.5 below, 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 9.3.** Show that $k + \Gamma \in \mathcal{K}_z$

So usually $\mathcal{K}_z = \{k, k+\Gamma\}$ and the attack in Attack 9.1 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 $k_{\text{guess}}$ 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 $k_{\text{guess}}$ 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 $k_{\text{guess}}$ if $x = 0100\,0000$? What if $x = 0000\,0000$? [*Hint:* use (1) and (2).]

**Exercise 9.4.**

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

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

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

**Exercise 9.5.** Let $\Gamma \in \mathbb{F}_2^8$ be non-zero. Show that for each non-zero $\Delta \in \mathbb{F}_2^8$,

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

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

Example 9.2 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.

**Lecture 28**

*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 8.4), the Feistel network sends $(v, w)$ to $\big(w, v + S(w + k^{(1)})\big)$ where

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

**Lemma 9.6.**

(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 9.7.** We run Attack 9.1 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

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

Trying each guess together with all 16 possibilities for $k_{\mathrm{guess}}^{(1)} \in F_2^4$ 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 9.6(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 $a$ is secret, but $2^a$ is 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 write $2^{ab} \bmod p$ in binary and take the 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: $c^{p-1} \equiv 1 \bmod c$ for any $c$ not divisible by $p$.

- If $c^m \not\equiv 1 \bmod p$ when $1 \le m < p-1$ 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 3 on Sheet 8. 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 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.

[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 $\mathrm{lcm}(t, t')$. 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 $t \ge p-1$. $\square$

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 $a = 5$ then we have $28 = 9 \times 3 + 1$ so

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

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

This example shows all the ideas needed for the proof of Lemma 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*

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

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

**Example 10.6.** Let $p = 11$, $q = 17$, so $n = pq = 187$ and $(p-1)(q-1) = 160$. Let $a = 9$. 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 $a$, $p$ and $q$ it is easy to find $r$ as in Fact 10.5. But it is believed to be hard to find $r$ given only $a$ and $n$. If so, $x \mapsto x^a \bmod n$ is a one-way function, suitable for use as the encryption function in a cryptosystem.

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

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

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

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

and

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

The encryption functions are defined by

$$e_a(x) = x^a \bmod n.$$

Alice's *public key* is the pair $(a, n)$. In private Alice computes $r$ such that $ar \equiv 1 \bmod (p-1)(q-1)$. As just seen, she can do this because she knows the key, so she knows $(p-1)(q-1)$. The decryption function is then

$$d_a(y) = y^r \bmod n.$$

Alice's *private key* is the pair $(r, n)$.

**Example 10.7.**

(1) For a small example, take $p$ and $q$ as in Example 10.6. If Alice's public key is $(9, 187)$ then her private key is $(89, 187)$. 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 `PKCExamples.nb` available from Moodle can be used to give examples where $p$ and $q$ are large.

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 it into $p-1$ and $q-1$. Then $x^a \bmod n$ can be computed quickly by repeated squaring.

The demonstration in the lecture went wrong for an interesting reason: by mistake the message $x$ was more than $n$, so $x$ was not a permitted plaintext. After decrypting the received message was $x^{ar} \equiv x \bmod n$, not $x$ itself.

Question 3 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 $n$. (At least no-one has found such an attack.)

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.

*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.

## 11. Digital signatures and hash functions

In this section we suppose the possible messages are elements of $\mathbb{N}_0$. Using the ASCII encoding (see Question 4 on Sheet 5), any English message can be put in this form.

*Digital signatures.* Suppose Alice and Bob have RSA keys:

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

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

Malcolm 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.

[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.

To do this, Malcolm must know Alice's public key. So the attack is specific to public key cryptosystems such as RSA. If there is no public key then only Alice and Bob know the encryption function $e_k$.

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 $r$ is a function $h : \mathbb{N}_0 \to \mathbb{F}_2^r$. The value $h(x)$ is the *hash* of the message $x \in \mathbb{N}_0$.
(b) Let $(b, n)$ be Bob's public key. 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 $r$ behaves like a random function from $\mathbb{N}$ to $\mathbb{F}_2^r$. 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 $\frac{1}{2^r}$. Hence the number of trials until first success is distributed geometrically with parameter $\frac{1}{2^r}$, so on average $2^r$ trials are needed. Thus in (b) 'hard to find' means 'requires at least $2^r$ hashes'.

**Exercise 11.3.** Let $h : \mathbb{N} \to \mathbb{F}_2^r$ 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 $B = 2^r$. Since $\sqrt{2^r} = 2^{r/2}$ we interpret 'hard to find' as 'requires at least $2^{r/2}$ hashes'.

*Hash functions in practice.* We have already seen one way to make a hash function. Fix a block cipher of length $r$ and a key $k$. Chop the message $x$ into blocks $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, such that each $x^{(i)} < 2^r$. Let $b^{(i)} \in \mathbb{F}_2^r$ be the binary form of $x^{(i)}$. Then apply the block cipher in cipher block chaining mode (see page 41), 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 $y^{(t)} \in \mathbb{F}_2^r$ 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?

**Example 11.7** (Bitcoin blockchain)**.** 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 $\big(x, d_a(h(x)\big)$.

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 make another $\big(x', d_a(h(x'))\big)$ where $x'$ says 'I Alice give Charlie the bitcoin $b$'.

To avoid this, transactions are *validated*. To validate a list 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 this list is converted to a number, its hash, by two iterations of SHA-256, has a large number of initial zeros. 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.

[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.

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.[21]

On the day of writing (8th December 2017) the bitcoin is at a record high[22] of \$13069.41 and the reward for growing the blockchain is 12.5 bitcoins. (This gradually decreases; there will never be more than $21 \times 10^6$ bitcoins in circulation.) Most transactions therefore involve small fractions of a bitcoin. A typical block verifies about 2500 separate transactions.

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 visit 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]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; (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.

[22]Update one day later: it is now \$16063.86