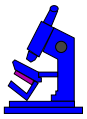
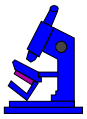



Chapter 6

Lecture 6 - Errors


Assignment

- ✔ Form your groups,
- ✔ Select your project,
- ✔ Email me with your proposal for approval

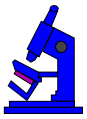


Mid semester Test

- ✔ 9th October 2003
- ✔ LT27, 14:30
- ✔ MCQ, closed book
- ✔ Covers everything up to the lecture before...

Last session

- Channel properties
- Entropy
- Models



This session

- Information flow
- Simple error detection
- Simple error correction
- Encryption

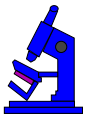
Information flow (Chapter 16)

✔ We may also more abstractly model some security policies by considering the flow of information in a system.

✔ We can use *entropy* to formalize this.

✔ In this context, we can establish quantitative results about information flow in a system, rather than just making absolute assertions[5].

[5] For example, "*System X reveals no more than 25% of the input values*".



Information flow

In the textbook we have a definition of information flow based on the conditional entropy $H(x \mid y)$ of some x given y:

Definition 16-1. The command sequence c causes a flow of information from x to y' if $H(x \mid y') < H(x \mid y)$. If y does not exist in s then $H(x \mid y) = H(x)$.

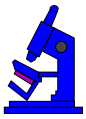
We can use this to detect *implicit* flows of information, not just explicit ones in which we directly modify an object.

Information flow

Consider the example on page 409 of the textbook:

```
if x=1 then
   y := 0
else
   y := 1;
```

After this code segment, we can determine if $x = 1$ from y' even though we do not ever assign y' directly from some function of x. In other words we have an implicit flow of information from x to y'.



Information flow

Formal treatment by considering the entropy of x. If the likelihood of $x = 1$ is 0.5, then $H(x) = 1$. We can also deduce that $H(x \mid y') = 0$, and so

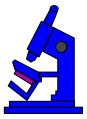
$$H(x \mid y') < H(x \mid y) = H(x) = 1$$

and information is flowing from x to y'. Paper gives some background.

Confinement and covert channels

The confinement problem is one of preventing a system from leaking (possibly partial) information.

Sometimes a system can have an unexpected path of transmission of data, termed a covert channel, and through the use of this covert channel information may be leaked either by a malicious program, or by accident.



Confinement and covert channels

Consider the set of permissions on a file.

An unscrupulous program could modify these permissions cyclically to transmit a very-low data-rate message to another unscrupulous program.

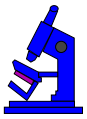
Confinement and covert channels

We categorize covert channels into two:

1. **Storage channels:** using the presence or absence of objects

2. **Timing channels:** the speed of events

We can attempt to identify covert channels by building a shared resource matrix, determining which processes can read and write which resources.



Attacks on databases

- ✔ Governing body may keep secret individual information, but release *cumulative* information
- ✔ For example: Today's average temperature of SOC staff by nationality:

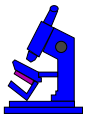
Singaporean	Malaysian	PRC	Poland	German	Australian	New Zealand	....
36.8	36.7	36.9	37.1	36.5	38.2	38.1	....

Attacks on databases

- ✔ OK - doesn't release any sensitive information, but
- ✔ what if another part of the database released the numbers of SOC staff by nationality...

Singaporean	Malaysian	PRC	Poland	German	Australian	New Zealand	....
23	12	14	3	5	4	1	....

- ✔ By inference you can deduce that the temperature of a particular individual is too high!



This session

- Information flow
- Simple error detection
- Simple error correction
- Encryption

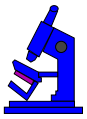
Simple check codes

✔ Transmit data:

1	65	3	22	47	2

✔ Transmit data+checksum:

1	65	3	22	47	2	140

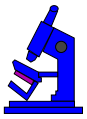



One-way parity

A	0	1	0	0	0	0	0	1
0	0	0	1	1	0	0	0	0
D	0	1	0	0	0	1	0	0
B	0	1	0	0	0	0	1	0
B	0	1	0	0	0	0	1	0
C	0	1	0	0	0	0	1	1
Check:	0	1	1	1	0	1	1	0


Two way parity

A	0	1	0	0	0	0	0	1	0
0	0	0	1	1	0	0	0	0	0
D	0	1	0	0	0	1	0	0	0
B	0	1	0	0	0	0	1	0	0
B	0	1	0	0	0	0	1	0	0
C	0	1	0	0	0	0	1	1	1
Check:	0	1	1	1	0	1	1	0	X



Simple check codes

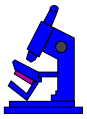
- ✔ Parity of bits - detects all 1 bit errors, but...
- ✔ Horizontal and vertical parity - better, but problems with repetitive errors
- ✔ Sum of values - problems with repetitive errors
- ✔ Want better level of error checking

Cyclic redundancy check codes

Treat the stream of transmitted bits as a representation of a polynomial with coefficients of 1:

$$10110 = x^4 + x^2 + x^1 = F(x)$$

Checksum bits are added to ensure that the final composite stream of bits is divisible by some other polynomial $g(x)$.



Cyclic redundancy check codes

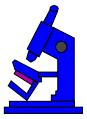
- ✔ We can transform any stream $F(x)$ into a stream $T(x)$ which is divisible by $g(x)$.
- ✔ If there are errors in $T(x)$, they take the form of a difference bit string $E(x)$ and the final received bits are $T(x) + E(x)$.
- ✔ When the receiver gets a correct stream, it divides it by $g(x)$ and gets no remainder.

Cyclic redundancy check codes

The question is: *How likely is that $T(x) + E(x)$ will also divide with no remainder?*

Single bits? - No a single bit error means that $E(x)$ will have only one term (x^{1285} say). If the generator polynomial has $x^n + ... + 1$ it will never divide evenly.

Multiple bits? - Various generator polynomials are used with different properties. Must have one factor of the polynomial being $x^1 + 1$, because this ensures all odd numbers of bit errors (1,3,5,7...).



Some common generators:

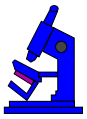
- **CRC-12** - $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
- **CRC-16** - $x^{16} + x^{15} + x^2 + 1$
- **CRC-32** - $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+1$
- **CRC-CCITT** - $x^{16} + x^{12} + x^5 + 1$

Long division is easy!

Generator $g(x)$: $x^5 + x^2 + 1$ (100101) and $F(x)$: 101101011. divide $F(x)$ by $g(x)$, and the remainder is appended to $F(x)$ to give $T(x)$:

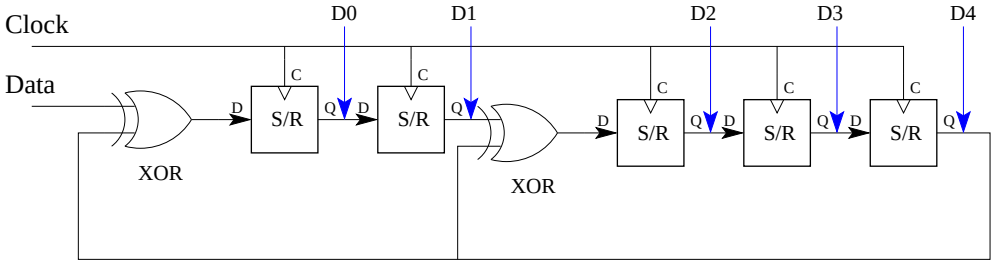
```
            1010.01000
100101 )101101011.00000
        100101
         100001
          100101
             1001.00
              1001.01
                    1000
```

$T(x) = 10110101101000.$



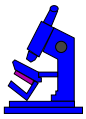
Long division is easy!

When this stream is received, it is divided but now will have no remainder if the stream is received without errors.



Long division is easy!

Input data	D4	D3	D2	D1	D0	Note
...	0	0	0	0	0	Initial state
1	0	0	0	0	1	First bit
0	0	0	0	1	0	Second bit
1	0	0	1	0	1	Third bit
1	0	1	0	1	1	
0	1	0	1	1	0	
1	0	1	0	0	0	
0	1	0	0	0	0	
1	0	0	1	0	0	
…						



Long division is easy!

Input data	D4	D3	D2	D1	D0	Note
...						
1	0	1	0	0	1	
0	1	0	0	1	0	
0	0	0	0	0	1	
0	0	0	0	1	0	
0	0	0	1	0	0	
0	0	1	0	0	0	

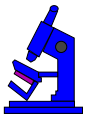
Case study: ethernet

Ethernet is used for networking computers, principally because of its speed and low cost. The maximum size of an ethernet frame is 1514 bytes[6], and a 32-bit FCS is calculated over the full length of the frame.

The FCS used is:

- **CRC-32** - $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+1$

[6] 1500 bytes of data, a source and destination address each of six bytes, and a two byte type identifi er. The frame also has a synchronizing header and trailer which is not checked by a CRC.



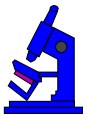
This session

- Information flow
- Simple error detection
- Simple error correction
- Encryption

Simple error correction

Methods used to correct errors:

- Ignore errors, while acknowledging correct data. ***ARQ*** (for **A**utomatic **R**epeat re**Q**uest).
- Error correcting codes (for computer memory)



Code types

We can divide error correcting codes (ECC) into continuous and block-based types. Convolutional encodings are used for continuous systems, and the common block-based codes are:

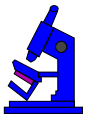
- Hamming codes (for correcting single bit errors),
- Golay codes (for correcting up to three bit errors), and
- Bose-Chaudhuri-Hocquenghem (***BCH***) codes (for correcting block errors).

Combining error correcting codes

✔ Different types of error correcting codes can be combined to produce composite codes.

✔ For example, *Reed-Solomon* block-codes are often combined with convolutional codes to improve all-round performance.

✔ In this combined setup, the convolutional code corrects randomly distributed bit errors but not bursts of errors while the *Reed-Solomon* code corrects the burst errors.

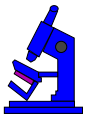


Accepting bad data

Sometimes we are willing to accept bad data...

BER and noise

System	Error rate (errors/bit)
Wiring of internal circuits	10^{-15}
Memory chips	10^{-14}
Hard disk	10^{-9}
Optical drives	10^{-8}
Coaxial cable	10^{-6}
Optical disk (CD)	10^{-5}
Telephone System	10^{-4}



BER and noise

We can determine the theoretical channel capacity knowing the SNR:

- BER is 0.01, channel capacity $C \simeq 0.92$ bits/symbol.
- BER is 0.001, channel capacity $C \simeq 0.99$ bits/symbol.
- BER is 0, channel capacity $C = 1$ bits/symbol.

The theoretical maximum channel capacity is quite close to the *perfect* channel capacity, even if the BER is high.

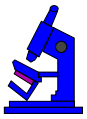
Reducing BER

✔ Increase the signal (power), or

✔ Reduce the noise (often not possible), or

✔ Use ECC.

The benefit of error correcting codes is that they can *improve* the received BER without increasing the transmitted power. This performance improvement is measured as a system ***gain***.



Reducing BER

Example: Consider a system without ECC giving a BER of 0.001 with a *S/N* ratio of *30dB* (1000:1). If we were to use an ECC *codec*, we might get the same BER of 0.001 with a S/N ratio of *20dB* (100:1). We say that the system gain due to ECC is *10dB* (10:1).

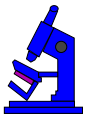
Bad ECC scheme: repetition

An initial scheme to correct transmission errors might be to just repeat bits[7].

```
Data:     0  1  0  0  1  1  1  1  ...
Transmit: 000111000000111111111111...
```

If we send three identical bits for every bit we wish to transmit, we can then use a voting system to determine the most likely bit. If our natural BER due to noise was 0.01, with three bits we would achieve a synthetic BER of 0.0001, but our channel capacity is reduced to about $C = 0.31$ bits/symbol.

[7] Note: there is no point in repeating bits *twice*. you must repeat three times, or 5 times, and then vote to decide the best value.

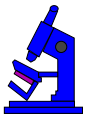


Bad ECC scheme: repetition

- ✔ We can see from this that the rate of transmission using *repetition* has to approach zero to achieve more and more reliable transmission.
- ✔ However we know that the theoretical rate should be equal to or just below the channel capacity C.
- ✔ Convolutional and other encodings can achieve rates of transmission close to the theoretical maximum.

ECC scheme: Hamming

- ✔ *Hamming* codes are block-based error correcting codes.
- ✔ We add hamming bits to a string
- ✔ Here we derive the inequality used to determine how many extra *hamming* bits are needed for an arbitrary bit string.



ECC scheme: Hamming

The *hamming* distance is a measure of how FAR apart two bit strings are.

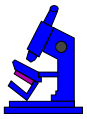
```
A:          0 1 0 1 1 1 0 0 0 1 1 1
B:          0 1 1 1 1 1 1 0 0 1 0 1
A XOR B:    0 0 1 0 0 0 1 0 0 0 1 0
```


ECC scheme: Hamming

If we had two bit strings X and Y representing two characters, and the *hamming* distance between any two codes was d, we could turn X into Y with d single bit errors.

- If we had an encoding scheme (for say ASCII characters) and the minimum *hamming* distance between any two codes was $d + 1$, we could ***detect*** d single bit errors[8].
- We can ***correct*** up to d single bit errors in an encoding scheme if the minimum *hamming* distance is $2d + 1$.

[8] Because the code d bits away from a correct code is not in the encoding.



ECC scheme: Hamming

If we now encode m bits using r extra *hamming* bits to make a total of $n = m + r$, we can count how many correct and incorrect *hamming* encodings we should have. With m bits we have 2^m unique messages - each with n illegal encodings, and:

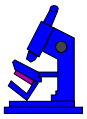
$$\begin{aligned} (n+1)2^m &\leq 2^n \\ (m+r+1)2^m &\leq 2^n \\ m+r+1 &\leq 2^{n-m} \\ m+r+1 &\leq 2^r \end{aligned}$$

ECC scheme: Hamming

We solve this inequality, and then choose R, the next integer larger than r.

Example: If we wanted to encode 8 bit values ($m = 8$) and be able to recognise single bit errors:

$$\begin{aligned} 8+r+1 &\leq 2^r \\ 9 &\leq 2^r - r \\ r &\simeq 3.5 \\ R &= 4 \end{aligned}$$



Reed-Solomon codes

- ✔ Reed-Solomon codes are block-based error correcting codes which are particularly good at correcting bursts (sequences) of bit errors.
- ✔ They are found in a wide range of digital communications and storage applications.
- ✔ Reed-Solomon codes are used to correct errors in digital wireless applications such as wireless LAN systems, and low Earth orbit (LEO) satellite communication systems.

Reed-Solomon codes

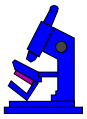
A Reed-Solomon code is specified as

- ***RS(n,k)*** with ***s***-bit symbols.

This means that the encoder takes k data symbols of s bits each and adds parity symbols to make an n symbol There are $n - k$ parity symbols of s bits each.

A Reed-Solomon decoder can correct up to t symbols that contain errors in a codeword, where

$$2t = n - k$$



Reed-Solomon code

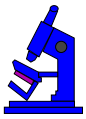
Example: A popular Reed-Solomon code is *RS(255,223)* with *8*-bit symbols. Each codeword contains 255 code word bytes, of which 223 bytes are data and 32 bytes are parity. In this example, $n = 255$, $k = 223$, and $s = 8$.

$$\begin{aligned} 2t &= 32 \\ \text{and so } t &= 16 \end{aligned}$$

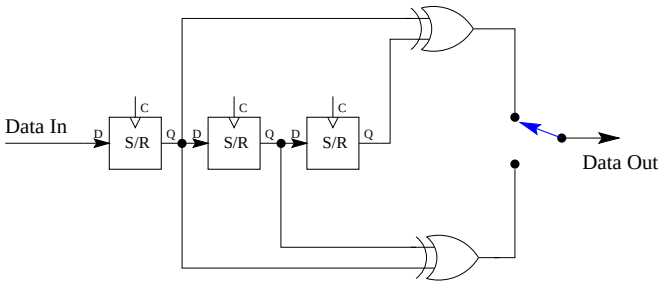
The Reed-Solomon decoder in this example can correct any 16 symbol errors in the codeword.

Convolutional codes

- ✔ Convolutional codes are designed to operate continuously and so are especially useful in data transmission systems.
- ✔ The convolutional *encoder* operates on a continuous stream of data using a shift-register to produce a continuous encoded output stream.



Convolutional codes



Received bit sequence can be examined for the *most likely correct* output sequence

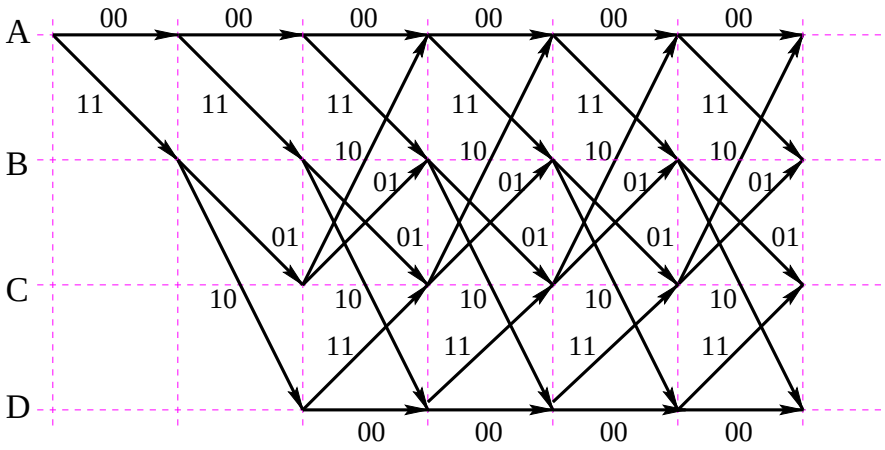
Convolutional codes

The length of shift register used for a convolutional code is known as the ***constraint length***, and it determines the maximum number of sequential input bits that can affect the output. The code rate R_{code} is the ratio of the input symbol size to output encoding size:

$$R_{\text{code}} = \frac{k}{n}$$

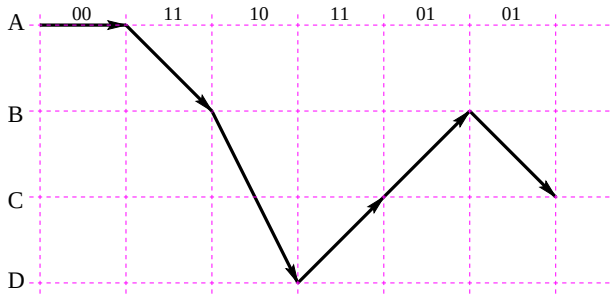
This coder produces two bits for every single bit of input, and the resultant tree of state changes repeats after three bits - that is, it only has four distinct states.

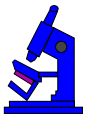
Trellis diagram



Most likely path

If we were to input the sequence **011010**, we would get the following trace through the trellis, with the bit sequence output as **001110110101**:



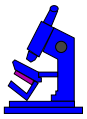


Convolutional codes

- ✔ Determine the *most likely* path, even with large numbers of bit errors.
- ✔ A rate $\frac{1}{2}$ convolutional encoding can often reduce errors by a factor of 10^2 to 10^3.

Viterbi decoding

- ✔ The ***Viterbi*** algorithm tries to find the most likely received data sequence, by keeping track of the four *most likely* paths through the trellis.
- ✔ For each path, a running count of the *hamming* distance between the received sequence and the path is maintained.
- ✔ The most likely received string is the one with the lowest hamming code.



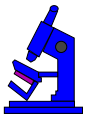
Case study: ECC encoders

Example: The COic5127A from Co-Optic Inc, contains a modern high data rate programmable Reed Solomon encoder that will encode blocks of up to 255 eight bit symbols to provide corrections of up to 10 errors per code block at data rates up to 320 Mbs. The output code block will contain the unaltered original data symbols followed by the generated parity symbols.

The chip supports encoding rates from 0 to 320 Mbs, and comes in a 68 Pin J leaded plastic chip carrier.

This session

- Information flow
- Simple error detection
- Simple error correction
- Encryption



Encryption and authentication

Security and Cryptographic systems act to reduce failure of systems due to the following threats:

Interruption - attacking the availability of a service (Denial of Service).

Interception - attacks confidentiality.

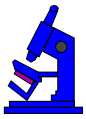
Modification - attacks integrity.

Fabrication - attacks authenticity. Note that you may not need to decode a signal to fabricate it - you might just record and replay it.

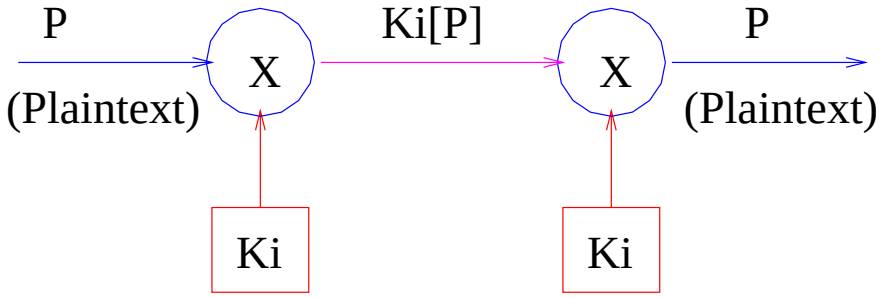
Encoding and deciphering

I could have told her the truth - that the same calculation which had served me for deciphering the manuscript had enabled me to learn the word - but on a caprice it struck me to tell her that a genie had revealed it to me. This false disclosure fettered Madame d'Urfé to me. That day I became the master of her soul, and I abused my power.

. We call these systems *symmetric* key systems.



Symmetric key systems

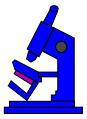


Simple ciphers - transposition

Transposition ciphers just re-order the letters of the original message. This is known as an anagram:

- *parliament* is an anagram of *partial men*
- *Eleven plus two* is an anagram of *Twelve plus one*

Perhaps you would like to see if you can unscramble "*age prison*", or "*try open*".

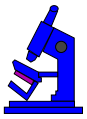


Transposition

- ✔ You can detect a transposition cipher if you know the frequencies of the letters, and letter pairs.
- ✔ If the frequency of single letters in ciphertext is correct, but the frequencies of letter pairs is wrong, then the cipher may be a transposition.
- ✔ This sort of analysis can also assist in unscrambling a transposition ciphertext, by arranging the letters in their letter pairs.

Simple ciphers - substitution

- ✔ Substitution cipher systems encode the input stream using a substitution rule.
- ✔ The Cæsar cipher is an example of a simple substitution cipher system, but it can be *cracked* in at most 25 attempts by just trying each of the 25 values in the keyspace.



Substitution

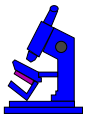
Code	Encoding
A	Q
B	V
C	X
D	W
...	...

If the mapping was more randomly chosen it is called a monoalphabetic substitution cipher, and the keyspace for encoding 26 letters would be $26! - 1 = 403,291,461,126,605,635,583,999,999$.

Substitution

If we could decrypt $1,000,000$ messages in a second, then the average time to find a solution would be about $6,394,144,170,576$ years!

We might be lulled into a sense of security by these big numbers, but of course this sort of cipher can be subject to frequency analysis.



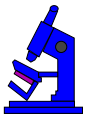
Frequency analysis

In the English language, the most common letters are: "E T A O N I S H R D L U..." (from most to least common), and we may use the frequency of the encrypted data to make good guesses at the original plaintext.

We may also look for *digrams* and *trigrams* (th, the).

One-time pad

If the key is large (say the same length as the text) then we call it a one-time pad.



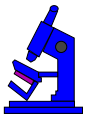
Vigenère

The Vigenère cipher is a polyalphabetic substitution cipher invented around 1520.

We use an encoding/decoding sheet, called a *tableau* as seen in Table 1, and a keyword or key sequence.

Vigenère

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**...**
A	A	B	C	D	E	F	G	H	...
B	B	C	D	E	F	G	H	I	...
C	C	D	E	F	G	H	I	J	...
D	D	E	F	G	H	I	J	K	...
E	E	F	G	H	I	J	K	L	...
F	F	G	H	I	J	K	L	M	...
G	G	H	I	J	K	L	M	N	...
H	H	I	J	K	L	M	N	O	...
...	...	...	...	...	...	...	...	...	...



Vigenère

If our keyword was BAD, then encoding HAD A FEED would result in

Key	B	A	D	B	A	D	B	A
Text	H	A	D	A	F	E	E	D
Cipher	I	A	G	B	F	H	F	D

If we can discover the length of the repeated key (in this case 3), and the text is long enough, we can just consider the cipher text to be a group of interleaved monoalphabetic substitution ciphers and solve accordingly.