The Playfair Cipher

The Playfair cipher is a simple but effective cipher that was actually invented not by Baron Playfair, but by the physicist and inventor Sir Charles Wheatstone. Playfair's main role was to popularize it. The British Foreign Office initially turned it down, because they thought it was too complicated. Wheatstone threatened to prove otherwise by teaching it to three out of four elementary school pupils in under 15 minutes, but the Foreign Office bureaucrat still thought it was going to be too hard for diplomats. In fact, it turned out to be easier to use than almost all the ciphers that were then in use, and it is strong enough that sixty years passed between its invention in 1854 and Joseph Mauborgne's publication of a method for its solution. It seems likely that cryptanalysts had been solving Playfair on the quiet for a while before that, but this is still an impressive achievement.
The idea is to arrange 25 letters in a five by five grid, as below. People usually leave out J, because it is rare, and you can get used to reading and writing I instead. Or you could cut out Q and keep J. Either way, you need 25 letters (or 36, 49,64…, but 25 is usual).

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

Letters are enciphered in pairs, and they need to be converted to upper case before this is done. So, if the message was "The office did not jump at the idea", the first step would be to split the message up into pairs, as follows TH EO FF … In fact, we need to stop there, because, for reasons that we'll see in a moment, Playfair requires the cipher clerk to break up pairs that are made of two copies of the same letter. To break up the pair, you add in an X between them. So the message actually comes out as

```
TH EO FX FI CE DI
DN OT IU MP AT TH
EI DE AX
```

Since there is an odd number of letters in the message, we add another X right at the end to make it come out even. Now we use the grid to encode the letters. The basic idea is to use the pairs of letters to define rectangles, then replace the plaintext letters by the ones at

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

]

the opposite corners of the rectangle. TH forms the rectangle shown above, so the first two letters of the ciphertext are SI. Notice that HT would give the same rectangle as TH, so we need to be careful about order: the rule is that a corner letter translates into the letter that is in the other corner of the same row.

EO forms another rectangle and yields DP. In the same way FX becomes HV.

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

The ciphertext we have so far is SI DP HV. You can probably see that we can (so far at least) get straight back to the plaintext by doing to it the exact same thing to the ciphertext that we have just done to the plaintext. We get the same rectangles as before, and the rules give us back the letters that we started out with. When we get to FI we find that F and I are in the same row, so the rectangle idea doesn't quite work.

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

In this case, the rule is that you encrypt by choosing the letter immediately to the right of the plaintext letter. So FI becomes GK. When you need to decrypt, you do the same, but you choose the letter immediately to the left of the ciphertext letter, so that you get back the plaintext. We say that the letter to the "right" of K is found by wrapping round to the beginning of the row and choosing F. In the same way, we say the letter to the "left" of V is Z. It's a shame that encryption and decryption are no longer exactly the same process, but they are still pretty similar.
The very next letter pair is CE, which works the same as FI, giving DA, when you remember to wrap around. Then we get DI, which poses the same problem as FI,, but this time in the vertical direction.

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

The solution is the same, but vertically. For encryption, you map the plaintext letters to the ones below them, wrapping around to the top if necessary. For decryption, you map the ciphertext letters to the ones just above them, wrapping around to the bottom if necessary, so DI becomes IO.

The ciphertext so far is SIDPHVGKDAIO (there's no need to be helpful to the cryptanalyst by actually printing the ciphertext in pairs). Carry on and you get

```
SIDPHVGKDAIO
CDTYKTNLDQCD
DKEACV
```

The traditional way of laying out ciphertext is in groups of five characters. This is left over from the days when messages were sent by telegraph in Morse code,

but we might as well respect tradition. The ciphertext (assuming I got it right) becomes:

```
SIDPH VGKDA
IOCDT YKTNL
DQCDD KEACV
```

Using Keys

We actually simplified a bit when we laid out the grid. Just listing the alphabet in the usual order is too obvious. In a way, the best thing to do would be to use a random alphabet. But we need the grid to be shared between the sender and the receiver. Random alphabets are difficult to memorize, and while you could write them down, the next thing that will happen is that they would get lost, dropped or even captured, allowing the enemy to read our messages. So what people do is to use a keyword, which should be something easy to remember. We'll use JIM TRESSEL as the keyword. Because we don't have J, replacing it with I, he actually comes out as IIM TRESSEL. You start by entering all the letters in the keyword, in order, leaving out repeats, then add in the remaining letters in alphabetical order, starting with A. (This is just the simplest way of forming the grid, cleverer methods are imaginable).

I	M	T	R	E
S	L	A	B	C
D	F	G	H	K
N	O	P	Q	U
V	W	X	Y	Z

Exercise 1

Try encoding the message GOBUCKS with the new grid. I think it comes out as FP CQ K U AV. You have to remember to pad the message with an X at the end so that it comes out even.

Exercise 2

Now make up a keyword, make up a message at least 20 characters long, write the message down in pairs, separate double letters, and encipher it according to the Playfair recipe.

I'm going to choose the keyword STZVLX, after the saint from Jasper Fforde's Thursday Next novels, The text I'm going to encode is "it remains, despite its non-existence, as one of the truly great wonders of Swindon", which is from Jasper Fforde's website.

In pairs, the message is:

```
IT RE MA IN SD
ES PI TE IT SN
ON EX IS TE NC
EA SO NE OF TH
ET RU LY GR EA
TW ON DE RS OF
SW IN DO NX
```

There turn out not to be any double letters , so the only X I had to add was the one at the end to make the pairs come out even.

My grid is

S	T	Z	V	L
X	A	B	C	D
E	F	G	H	I
K	M	N	O	P
Q	R	U	W	Y

The first two letters of my message are IT which turns into FL. Write the keyword as the first line of the message, in plaintext. Below that, write out the ciphertext of your message. For my one, the whole thing, split into fives, comes out as follows:

```
STZVLX
FLQFR FGPLX KXYPS FFLZK POKEE LSFOB FXVKK
GMHVF FSUWD LFUFX VRPOX IQTMH VQGPC PKB
```

I chose the keyword on purpose to include lots of letters from the end of the alphabet. This makes things a little harder for analysts who are hoping to see the usual pattern associated with keyword alphabets, which is that

Exercise 3

Swap messages with your neighbor and decipher your neighbor's message. You have the keyword, so this is mechanical.

Q. By the way, why did we choose to remove double letters?

Word segmentation.

When you decode a message that has used Playfair, you will find that there are some extra Xs which you have to read past. Also, you will have to find the word boundaries on your own, without help from spaces in the message. This is something which is not easy to mechanize (not impossible, I advised a Ph.D student working on computer models of how babies find word boundaries in spoken language). In the real world, telegraph operators make mistakes, so you also have to learn how to read past them, guessing what the sender really meant. Some errors are actually good for security, since they make life harder for the cryptanalyst without confusing the intended sender.

Summary of the rules for encoding using Playfair

1. Break the message up into pairs of letters. Double letter pairs are not allowed, so add an X to separate any pairs that would otherwise be doubles. If you have an odd letter left over at the end, add another X
2. If the letters in the pair are at the corners of a rectangle with at least two rows and two columns, then the ciphertext has the letters at the opposite corners of the rectangle. Choose the ciphertext letter that is in the same row as the corresponding plaintext
3. Otherwise, if the letters are in the same row, translate each of them as the next letter on the right. If you fall off the end of the row, wrap around to the beginning.
4. Otherwise, if the letters are in the same column, translate each of them as the next letter down the column. If you fall off the bottom of the column, wrap around to the top.

Summary of the rules for decoding using Playfair

1. If the letters in the pair are at the corners of a rectangle with at least two rows and two columns, then the ciphertext has the letters at the opposite corners of the rectangle. Choose the ciphertext letter that is in the same row as the corresponding plaintext
2. Otherwise, if the letters are in the same row, translate each of them as the next letter on the **left**. If you fall off the **beginning** of the row, wrap around to the **end**.
3. Otherwise, if the letters are in the same column, translate each of them as the letter **above**. If you climb off the **top** of the column, wrap around to the **bottom**.
4. Look at the resulting plaintext. Work out where the word boundaries are, and where the cipher clerk at the other end has added in extra Xs. This requires judgment, and is not completely mechanical.

Mnemonics for ciphering and deciphering using Playfair

The army field manual suggests learning and using the following two mnemonics to keep straight on the difference between the enciphering and the deciphering processes.

ERDL - Encipher right, decipher left.
EBDA- Encipher below, decipher above.

How to identify and break Playfair

This bit is a summary of material on Playfair from ch6 and ch7 of the Army Field Manual on cryptography. These are in the course web directory as

afm-ch6.pdf

afm-ch7.pdf

and you will also need to refer to the list of useful words in appendix D of the same field manual

appd.pdf

(don't print this out unless you definitely want to, it is long and boring). The chapters also explain some other polygraphic ciphers (Foursquare, Twosquare) that are roughly the same strength as Playfair. You don't need to know about these other ciphers for the course, but they might be interesting as background. The explanation of Playfair is really good, and you should study it carefully.

Facts about Playfair

- It is designed so that the monoalphabetic frequencies are concealed. E does not always translate to the same letter. This means that the frequency count will be smoother than a monoalphabetic cipher.
- The letters are encoded in pairs. This means that several measurable properties of the message are going to be even numbers. The length of the message will be. So will the lengths of any repeated sequences. So will the distances between repeats.
- Because of the coding in pairs, there are two different ways that any particular plaintext word can turn out. The word DODO will cause a repeat if it lines up as ... DO DO ... but it could equally be lined up as ... xD OD Oy ...
- While letter frequencies don't get preserved, the frequencies of **digraphs** (two letter pairs) do get preserved.
- The top few plaintext digraphs, in order of likely frequency are TH, IN, ER, RE, AN, HE, AR, EN, TI, TE, AT, ON, HA, OU, IT. Probably, the top few ciphertext digraphs will be translations of these.
- The most frequent ciphertext digraphs are very likely to be translations of plaintext digraphs that have the frequent letters E T A O I N in them. This helps because it gives us the connection between the frequencies of ciphertext digraphs and the (known) frequencies of plaintext letters.
- Playfair usually translates a letter by a letter from the same row. The only exception is when the same column rule kicks into play. So each plaintext letter can turn into just 5 different ciphertext letters, depending on what is around it. This means that the monoalphabetic frequency distribution is preserved to some extent.

- Every letter changes. All the Playfair rules produce a ciphertext letter that is different from the corresponding plaintext letter. This means that you can find the location of a possible word (also known as crib) by elimination. You can immediately rule out any place where the plaintext letter and the ciphertext letter would match.
- Patterns of digraph repetition are useful. If you look for words that have DO then an even number (including zero) of intervening letters, then DO again, this will correspond to a repeated digraph. Appendix D of the Army Field Manual has some words like this.
- The mirror image of a digraph uses the same rectangle as the digraph itself. This means that if plaintext ER is (say) UH, then plaintext RE will certainly be HU. This is useful when words like BATTALION line up as ... xB AT TA LI ... because the pattern of repetitions in the plaintext comes through in the ciphertext. Again, this pattern is potentially useful either when there is no separation between the pairs or when the separation is an even number of pairs.
- The only way that the plaintext version of a digraph can share a letter with the corresponding ciphertext digraph is if the letters of the digraph are next to each other in the grid (either horizontally or vertically, including wrapping around).
- It is possible to solve the message without getting the keyword. This is because of the wraparound. Once you have the message, you can rotate the rows or columns up-down and/or left-right and it will be the same cipher. If there is a keyword, eventually you will find it in the top left of the grid.

The field manual solution method

The field manual teaches a method for solving Playfair that relies on a **crib**. This is tried and tested, but rather tends to downplay the role of frequency analysis. In general, I recommend doing a frequency analysis for every message you work with, just as a matter of routine.

Homework (due next Monday 4 Feb)

Usual rules: decode this message. See if you can reconstruct the keyword. If you get stuck, showing your work carefully and precisely in the way DJ explained will get you full credit. The words “feathers”, “consensus” and “ungainly” appear in the plaintext. You can use them as cribs if it helps you. There are 774 characters, which is more than enough.

```
HEFCD HURLS DLZDA EEDOL YGOTO TWNSH EPFOG
METCO XHMDE DQDWN FPHSQ OQMZM HEMAH EHQMU
HCMWS DFBFA UKDTM GHPXH ZLQTL MKQHN HCEUH
THETC DSAPA YOQOC DSEYO TOTDS LESMA ZFYLB
YGPHU MVGDQ OFMAD OHTOT TGDQM WAZUT GMQOH
EDOPA IEMBQ TELHE OFPHO ADFQD FHHEF EHSQH
PEWHQ NGUTB XHFHH QUMOD FHPFZ MDTLQ PNWBG
MQFMA EDSME LHEDE HVODR HDSFH PFZMH FTHTB
VHUMS ZCOQD VMLBH VGZET COXHM DEDTY EPTBQ
DBAZL IRLOM HTEBU FEBAF OOHED UMYMP HPVNL
KNOHE DPLAL BIDLO FSLMA ONTSH FKNOT SQMPQ
OMAHV LGLOB WQSFQ DQHQE FHTPH MOLMH DHEWN
ABMOM BQTPL MZQOE GDSHE FQKNM OHPQT HVHED
TYMMB SLBXC KTFTR KNBAK ZGNBM HYBMP LSKCK
WPETI HNWHD TGRHM BQTOH EDWNA BRHUM YMPHP
VNWQB CBHDH QVHQH DUHCB DHOTO EZDSO TTGDQ
QOWNQ DWEYG DOHVP FTABP SIWBF ONUBM COMOU
LQHQM TZSQB ALEYM XDGMZ SMADQ AVHPH PMZQI
FOGMI ODHED NDOCL MHEFQ KNMOH SDHTZ QOHEW
NMBQT FHPFZ MHFFQ DSDHE DTYEP EHEDW TMDYA
XTQOT BTGGS TFGKI HNWHD OHEDW NABRH HVBCA
EFCHQ TEQOX HVNPA HKHRT HQOEU OATCD QUMQH
MHGZ
```