Parabola Volume 36, Issue 3 (2000)

Cryptography

PART 2

Rod James[1]

In the previous issue of Parabola[2] we saw how to encode (and also how to break!) monoalphabetic ciphers (i.e. we replace each letter of the alphabet by some other letter every time it occurs in the message). We now look at some more complex codes.

Polyalphabetic Codes

The insecurity of a monoalphabetic code is due to the fact that each time a given letter occurs in the original message, it is encoded using the same letter in the encrypted text. Thus, if the message is long enough, the code can be broken by comparing the frequency of each letter in the encypted text with frequencies of letters in most messages. For example, in the earlier article, we were able to recognize the encryption of the letters E and T because they were the most frequent and the frequency of the pair ZW suggested that it was the encryption of TH.

To overcome this insecurity, we (mentally) write the message in a number of columns and use different (monoalphabetic) codes for each column. The easiest way to do this is to choose a word (called the **keyword**), write the message in the same number of columns as the number of letters in the keyword and then use a "Caesar cipher" by adding to each column the number corresponding to that letter of the keyword. For example, if we chose the word CAT as the keyword then we would write the message in 3 columns and add 3 (mod 26) to the first column, 1 (mod 26) to the second column and 20 (mod 26) to the third column. Thus Veni Vidi Vici would be encrypted as follows:

$$\begin{array}{cccc} \text{VEN} & & 22 & 5 & 14 \\ \text{IVI} & \longrightarrow & 9 & 22 & 9 \\ \text{DIV} & & 4 & 9 & 22 \\ \text{ICI} & & 9 & 3 & 9 \end{array} \xrightarrow[(\text{mod } 26)]{\text{add } 3,1,20} \begin{array}{ccc} 25 & 6 & 8 \\ 12 & 23 & 3 \\ 7 & 10 & 16 \\ 12 & 4 & 3 \end{array} \longrightarrow \begin{array}{c} \text{YFH} \\ \text{LWC} \\ \text{GJP} \\ \text{LDC} \end{array}$$

which would then be written

Y F H L W C G J P L D C.

At first sight, it appears as though this code would be impossible to break. However, polyalphabetic codes can also be broken by using a little more ingenuity: all we need do is discover how many columns there are and then break the monoalphabetic code

[1] Rod James is a lecturer in Mathematics at UNSW and is the editor of Parabola.

[2] See Parabola Vol. 36 No. 2

for each column. For example, imagine that the following was confiscated from a (very silly — and not very smart) student during an examination:

CTAIF GBJGN QRUWF QNEZX ENQRN QVLMD ONFNR BHRFN QEYNV NRFNQ RMJZY JFNQR UBFCP AGNAN

where the letters are grouped in fives for convenience only. Note that the sequence NQR occurs 4 times, starting at positions 10, 22, 49 and 58, and so they are 12, 27 and 9 letters apart. Since these last three numbers are all multiples of 3, this suggests that a code was used with a keyword of length a multiple of 3. So we write the letters in triples (rather than columns for economy of space) as follows:

CTA IFG BJG NQR UWF QNE ZXE NQR NQV LMD ONF NRB HRF NQE YNV NRF NQR UBF CPA GNA N

Now notice that the pair NQ (letters 14 and 17 of the alphabet) occurs 6 times in the first 2 "columns" suggesting that this represents TH (letters 20 and 8 of the alphabet) in the original message. Since $14-20 \pmod{26}$ is 20 and $17-8$ is 9, try subtracting 20 (mod 26) from the first column and 9 (mod 26) from the second column. The result (with decrypted letters in lower case) is

ik A ow G ha G th R an F we E fo E th R th V rd D ue F ti B
ni F th E ee V ti F th R sa Z ea F th R as F ig A me A t.

It is now easy to see that the keyword was TIM and the message was
"I know that the answer for the third question is three. It is the same as the assignment".

Public Key Cryptography

All of the codes we have considered so far rely on secrecy for their security: if a spy who knows the code reveals it to anyone, then they can decipher any message and the code becomes useless. However, in 1976, Diffie and Helman suggested a new approach in which everyone knows the key required to encode a message (and so can do so), but a second key is needed to decode messages. This decoding key is only provided to those with authority to decode, and it is impossible (at least in a reasonable amount of time) to find the decoding key from a knowledge of the encoding key.

One of the earliest, and still widely used, public-key codes was created in 1977 by Rivest, Shamir and Adleman, now referred to as RSA-coding. To understand the RSA-code, remember that, if we have n symbols (e.g. $n = 26$ for the alphabet), then the Caesar cipher is an example of an "addition cipher" $m \to m + k (\mathrm{mod}\ n),$ where k is the key. Also it was suggested in the previous issue that a "multiplication" cipher $m \to km (\mathrm{mod}\ n)$ could be used provided that k is coprime to n. A more sophisticated cipher would be a "power" cipher $m \to m^k \,(\mathrm{mod}\ n),$ and a message encoded by such a cipher can be decoded using the following two results:

Theorem 1 *If a, b are two coprime integers, then there are integers x, y such that*

$$ax + by = 1.$$

Theorem 2 *If a is an integer and $\phi(n)$ is the number of integers between 0 and n which are coprime to n, then*

$$a^{\phi(n)} = 1 \pmod{n}.$$

Thus, if k is the key for a power cipher and k is coprime to $\phi(n)$, then there are integers x, y such that $kx + \phi(n)y = 1$ and so

$$m = m^1 = m^{kx+\phi(n)y} = m^{kx}m^{\phi(n)y} = m^{kx} \times 1 = m^{kx} \pmod{n}.$$

So decoding consists of raising the encoded message $c = m^k$ to the x'th power: $c \to c^x = (m^k)^x = m$. We will write $k^{-1} \pmod{\phi(n)}$ for this x.

Examples

1. Suppose we were encoding a message which included spaces between words, commas and fullstops (as well as the 26 letters of the alphabet) by replacing A by 1, B by 2, $\ldots$, Z by 26, space by 27, comma by 28 and fullstop by 0, and then using

   $$m \to m^3 \pmod{29}.$$

   Since all of the numbers $1, 2, \ldots, 28$ are coprime to 29,

   $$\phi(29) = 28 \qquad \text{and} \qquad 19 \times 3 - 2 \times 28 = 1.$$

   So $3^{-1} \pmod{28} = 19$ and decoding consists of $c \to c^{19} \pmod{29}$.

2. Suppose we only used the 26 letters of the alphabet (replacing letters by numbers as in example 1) and then used

   $$m \to m^5 \pmod{26}$$

   The set of numbers between 0 and 26 which are coprime to 26 is $\{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$ and so $\phi(26) = 12$. Also

   $$\phi(26) = 12 = 2 \times 5 + 2 \quad \text{and} \quad 5 = 2 \times 2 + 1.$$

   So

   $$1 = 5 - 2 \times 2 = 5 - 2(12 - 2 \times 5) = 5 \times 5 - 2 \times 12 \equiv 5 \times 5 \pmod{12}$$

   and $5^{-1} \pmod{12} = 5$. Decoding consists of $c \to c^5 \pmod{26}$.

In general, if p and q are prime numbers, then $\phi(p) = p - 1$ and

$$\begin{aligned}\phi(pq) &= pq - 1 - (\text{number of multiples of } q + \text{number of multiples of } p)\\ &= pq - 1 - (p + q)\\ &= (p-1)(q-1).\end{aligned}$$

Now the important feature of numbers of the form $n = pq$ is that it is impossible to calculate $\phi(n)$ unless you know p and q. Currently there is no way of quickly calculating the factors of large numbers (even by computer) and so a message which was encoded by $m \to m^k \pmod n$ cannot be decoded unless the factors of n are known, **even though n and k are known**. Thus a code of this form is safe even if everyone knows n and k provided they do not know p and q (or, equivalently $\phi(n)$) and so, we can create an RSA code, by doing the following:

1. choose 2 random large prime numbers p, q (say 100 digits), find $n = pq$ and calculate $\phi(n) = (p-1)(q-1)$;

2. choose a random number k between 1 and $\phi(n)$ which is coprime to $\phi(n)$ and find $\ell = k^{-1} \pmod{\phi(n)}$ using theorem 1.

The public key is then the pair (n, k) and the secret key is ℓ (where, now that ℓ has been found, we can forget p, q and $\phi(n)$).

Example 3 The problem with examples 1 and 2 is that the letter A can always be recognised since $1^k = 1$. So we use the following replacements:

$$0 \to 0, \quad 1 \to 1, \quad \text{space} \to 2, \quad \texttt{A} \to 3, \ \ldots, \texttt{Z} \to 28,$$

Now suppose that $p = 31$ and $q = 47$. Then $n = pq = 1457$ (public) and $\phi(n) = 30 \times 46 = 1380$.

If the public key is $k = 7$, then the secret key is $7^{-1} \pmod{1380} = 1183$.

We must keep p, q and $\phi(n)$ secret (or just forget them once 7^{-1} is found).

If anyone wanted to encode the letter Y, they would use the following:

$$\texttt{Y} \to 27 \to 27^7 \pmod{1457} = 914 = c.$$

If (and only if) someone knew the secret key, they could decode this as follows:

$$914 \to 914^{1183} \pmod{1457} = 27 \to \texttt{Y}.$$

This example is quite unrealistic since

(a) 1457 is easily factored and hence someone can easily break it;

(b) It is also simply a 29 letter monoalphabetic substitution and so is easily broken by the statistical methods above.

We could improve on (b) by encoding, and then enciphering, **pairs** of letters (a, b) by writing (a, b) as $29a + b$, as in the following example

$$\texttt{UP} \to (23, 18) \to 23 \times 29 + 18 = 685$$

then

$$685 \to 685^7 \equiv 1102 \pmod{1457} = c.$$

However, even this is still only a $29^2 = 841$ letter simple alphabetic substitution.

In real life, p and q are chosen to have about 100 digits each and so n has about 200 digits. This allows us to break the message up into 75-character chunks of integers and apply RSA to each chunk in turn. This is so large that statistical methods are useless for breaking it.

Also RSA is slow, since calculating powers mod n for large n needs special computer packages and lots of arithmetic. Hence RSA is rarely used in real life to send the *whole* message. What usually happens is:

1. A randomly generates a large *temporary* key k,
2. A encrypts the key k using RSA,
3. A encrypts the message m to c using k,
4. A sends both c and the encrypted key k to B,
5. B finds k using RSA decryption.
6. B then uses this k to decode the message.

The key k cannot be found by statitistical means since a different key is generated for each message.

If you are interested in learning more, you can start with "The Code Book" by Simon Singh (published by Fourth Estate) or visit the RSA webside `www.rsasecurity.com`.