Gerald's Column
by Gerald Fitton

In my last four articles I have described some of the properties and theorems of Modulus Algebra. This month I shall tell you about the famous RSA System.

Primes

In case you missed it last month I have improved the [c_Prime] custom function so that it now runs much faster than the original. This custom function returns the largest prime factor of any number within the range of integers available to the Floating Point Emulator.

My custom function is a poor way of finding a ***set*** of prime numbers because it starts with a clean sheet every time. A better method would be to create a store of prime numbers (as they are discovered) and then use that store when finding the next larger one. However, as a crude but simple check on whether a ***single*** number is prime it will suffice.

The Inventors

The famous RSA Public Key Encryption System was the invention (or should I say discovery?) of the trio Ronald Rivest, Adi Shamir and Leonard Adleman of MIT. They first published their findings in the August 1977 issue of Scientific American.

The Files

You will find all the files relevant to this article on the Archive monthly disc. They are available in both PipeDream and Fireworkz format in the RSA subdirectory. Let me remind you that the Fireworkz format files will load into the demo version of Resultz. Although I sent a copy of the Demo version of Resultz to Paul for inclusion on an earlier (monthly) disc there was insufficient space. It does appear on the Archive CD.

If you have these files from the monthly disc then double click on the file [TestKey] and the remainder of the set, five files in all, will load. The other four are custom functions.

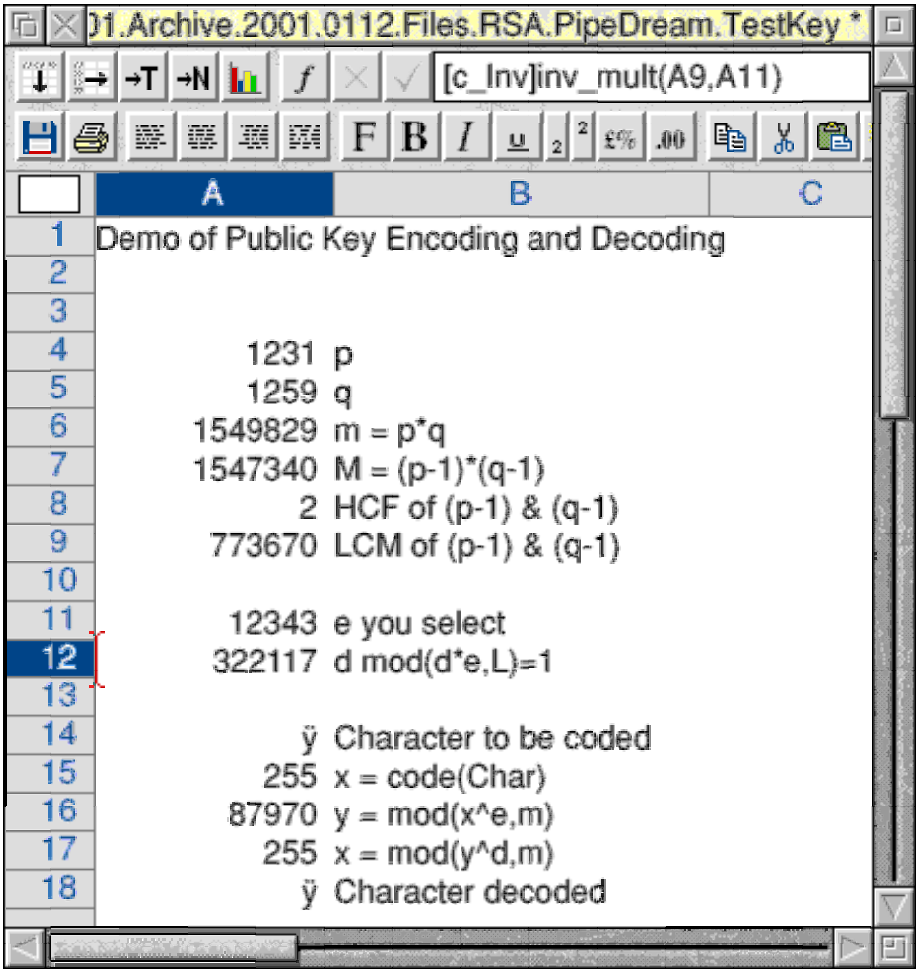
Just an Example

Please note that the size of the numbers which can be entered into my [TestKey] spreadsheet is limited by the precision of the Floating Point Emulator. It is intended to demonstrate principles; it is not meant to be a means of finding usable secure keys.

An Aside

The first screenshot below is of the PipeDream version of the file [TestKey]. As an aside, and in in response to suggestions from my correspondents, I have used Homerton rather than System font—and please, those of you with an older version of PipeDream, note that the characters in the formula line appear in Homerton Font.

The second screenshot is of the Fireworkz version of the same file. As you will see, with some versions of the OS, the font used in the formula line is still the System Font.



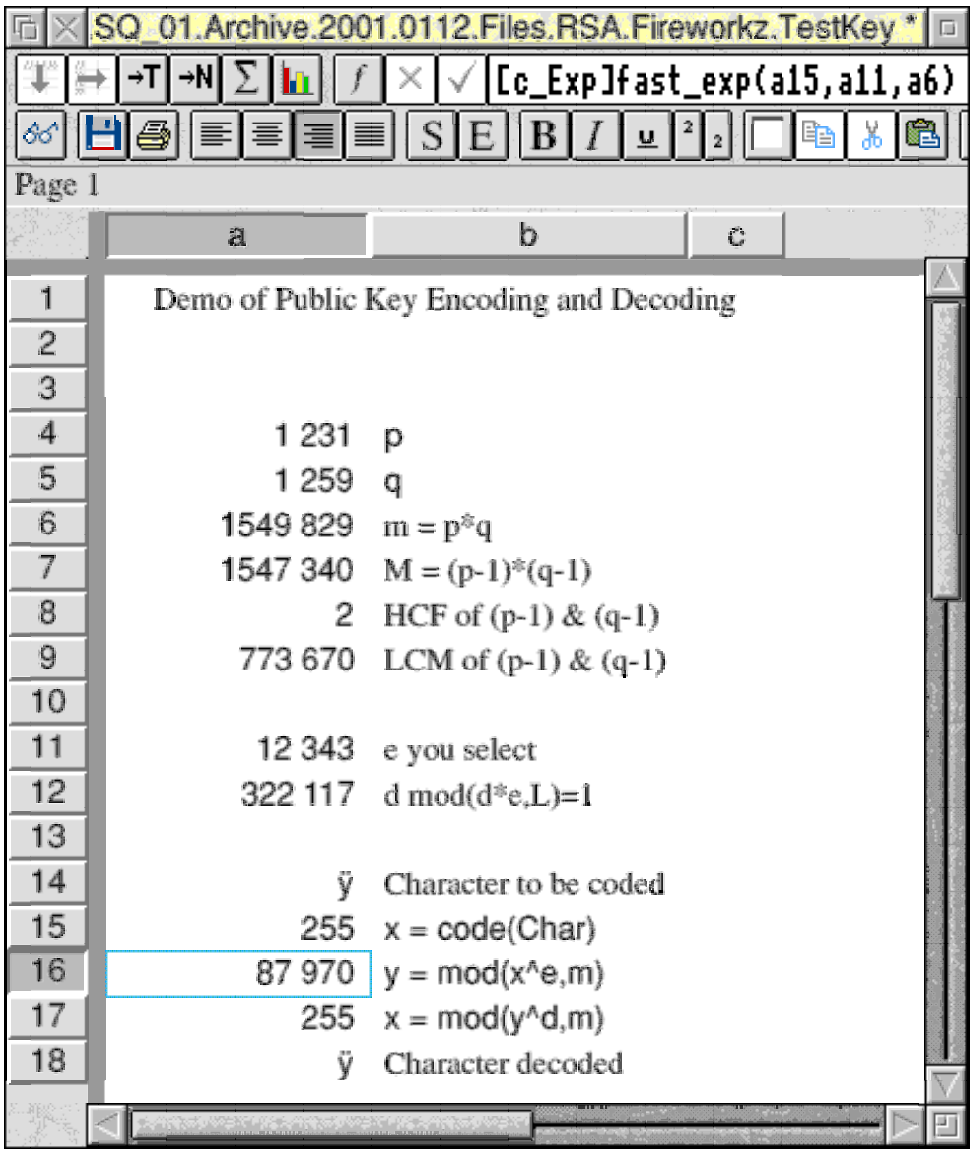
The RSA Protocol

Last month I described the DHM Protocol in detail. The RSA Protocol differs from it in many respects. Here are a few of those differences.

The potential user of the RSA System has the Key creation process totally under their own control—well, perhaps that is an overstatement because it is almost certain that the potential user will be provided with a computer program which can be used to create a pair of Keys. The Protocol for setting up a DHM exchange requires that the two participants exchange the values of ‘g’ and ‘m’ openly in advance and then each chooses an ‘e’ or a ‘d’ as their secret contribution to the Protocol.

The DHM Protocol allows the two participants to share a secret exchange. The Public Key feature of the RSA Protocol allows strangers to use the Public Key of a potential recipient to send them a secret message.

Another difference is that the Keys generated in a DHM exchange are generally used only once whereas RSA Keys are used many times.



Join the Club

From what I have written in the previous section you will appreciate that publishing your own RSA Public Key is rather like joining an existing club. When you have created your pair of Keys you send your Public Key to the club for publication in a directory. In return you receive the Public Keys of other members of the club.

Back in 1992 such a 'club' existed for Archimedes users! Unfortunately I can't find my copies of the club magazine but somewhere (on an unlabelled disc) I do have a Public and Private Key generated using BBC BASIC. If any of you can locate your copies then I shall be interested in hearing from you.

Almost certainly, whatever 'club' you aspire to belong to, you will use a computer program provided by the club to create your RSA Keys—my example is intended to demonstrate the principles which are embodied in such a program, not to replace it.

‘p’ and ‘q’

As a potential new member you choose (or use the computer program to generate) a couple of prime numbers. In my example these are shown in the screenshots above as ‘p’ and ‘q’.

The value of ‘p’ appears in [TestKey]A4. If you enter a number which is not prime into [TestKey]A4 then the error message “Not Prime” will appear in [TestKey]B4 rather than ‘p’. You will get the same error message in [TestKey]B5 if the number you choose for [TestKey]A5 is not prime.

The spreadsheet (or your club’s program) then does a few sums to calculate the values which appear in the block [TestKey]A6A9 (see the screenshot).

The value of m = p*q becomes the modulus of the one way functions used to encode and decode the messages which (you hope) will be sent to you by other members of the club.

Leonhard Euler

As another digression allow me to write a few words about the Swiss Mathematician Leonhard Euler (1707 – 1783). He was the ‘inventor’ (or ‘discoverer’) of the concept I have called M. In my example M = (p − 1)*(q − 1).

Leonhard Euler was fond of using Greek Symbols. Indeed it was he who chose the symbol π for pi and introduced the Γ (gamma) function to Mathematics. He chose the Greek letter Φ(m) for the number of numbers less than ‘m’ which are relatively prime to ‘m’. Let me remind you that a pair of ‘relatively prime’ numbers means that the HCF of the pair is one. Leonhard Euler proved many theorems about integers which involve this Φ(m) number. The number I have called M is what Leonhard Euler called Φ(m).

Many Mathematicians whose strengths are/were in Number Theory (mainly Integers) are less noted for their contributions to the Theory of Real Numbers. Leonhard Euler is an exception. His treatise on Differential and Integral Calculus were the standard textbooks and reference works for a century. He is the creator of the Calculus of Variations.

Φ(m)

Because ‘p’ and ‘q’ are prime numbers there are Φ(m) = M = (p − 1)*(q − 1) numbers less than ‘m’ which are relatively prime to ‘m’. I am not going to prove this to you. I’m sure that if you are interested enough you will be able to work it out for yourself.

The Public and Private Keys

The third and last number over which you have some control is ‘e’. In [TestKey]A12 my spreadsheet calculates the multiplicative inverse (of ‘e’), ‘d’, using the algorithm to which I first introduced you a couple of months ago. This algorithm is based on an expanded version of Euclid’s algorithm for finding the HCF of two numbers; it is embodied in my custom function [c_Inv].

The two values ‘e’ and ‘m’ are the Public Key. The values ‘d’ and ‘m’ are the Private Key.

If you choose an inappropriate value for ‘e’ then the error message “Bad Choice” will be returned in [TestKey]A12 instead of a value for ‘d’. A necessary criterion for ‘e’ is that it must be relatively prime to L, the modulus.

If ‘e’ is not relatively prime then the essential feature, one-to-one correspondence between ‘x’ and ‘y’ (coded and decoded character sets), will not exist.

M or L

It gives me great pleasure to acknowledge Colin Singleton’s contribution to this next section of my article.

In some expositions of the RSA System the number used in the formula of [TestKey]A12 is not L but M. Indeed, the exposition which is given in Sarah Flannery’s book, “In Code” doesn’t mention L. Colin refers to “The Public Key”, April 1992, a magazine edited by George Foot—the “F” of the BCF Encryption System. Colin remarks that on one page the magazine gives as the essential requirement “d and e are multiplicative inverses relative to M” and on another page “d and e are multiplicative inverses relative to L”. It is interesting that the two values of ‘d’ which are the result of using either M or L both seem to ‘work’. Colin has found a whole series of values of ‘d’ which ‘work’.

Because of my exchange of letters with Colin (of which the above is only a small part) I have decided to use L rather than M in my example. I would ask you to notify me (and Colin) if you find that there are circumstances in which M ‘works’ and L does not!

Colin also points out that the Key Space is maximised if L is made as large as possible for a given ‘size’ of ‘m’. This will be achieved if the HCF of (p − 1) and (q − 1) is 2. In support of this suggestion from Colin I quote a remark attributed to Ronald Rivest (the R of RSA) that both (p − 1) and (q − 1) should contain a large prime factor. Because ‘p’ is prime the number (p − 1) must be divisible by 2. Colin suggests that, in order to maximise L (and hence the Key Space) (p − 1)/2 and (q − 1)/2 should be prime as well as ‘p’ and ‘q’.

I mentioned that Colin found a whole series of values of ‘d’ which ‘work’. Colin points out that this series of possible values for ‘d’ has the fewest members if ‘p’ and ‘q’ are chosen so that (p − 1)/2 and (q − 1)/2 are both prime.

Encoding & Decoding

The encoding function is y = (x^e) mod m. You will see that it uses the two components of the Public Key, ‘e’ and ‘m’. I have used the ‘fast exponentiation algorithm’ in cell [TestKey]A16 to encode the character of A14. Again I must thank Colin for his help with the development of this ‘fast exponentiation algorithm’ custom function, [c_Exp].

The decoding function is x = (y^d) mod m. You will see from the screenshots that it uses the two components of the Private Key, ‘d’ and ‘m’. By now you will have guessed that this decoding function also uses the ‘fast exponentiation algorithm’.

Security

If you were a code breaker then you would know the values of ‘e’ and ‘m’ and you would know the nature of the one way modulus functions used by the RSA System. Furthermore you would have access to all sorts of fast multiplicative inverse and fast exponentiation computer routines.

In order to crack the code you need to be able to find L. If you can find L then you can find ‘d’ because it is the multiplicative inverse of ‘e’, modulus L.

Although Mathematicians do not agree on the best algorithms for finding the multiplicative inverse and for fast exponentiation, all Mathematicians who have studied the RSA System agree that, to crack the code, it is necessary to find ‘p’ and ‘q’ by factorizing ‘m’.

There are very clever methods for finding prime numbers and for finding the prime factors of numbers such as ‘m’. Some types of prime number are easier to find than other types. Computer programs which are used to select the values of the two initial prime numbers, ‘p’ and ‘q’, tend to use a particular sort of prime number and this may be a weakness.

The security of the RSA system relies upon the fact that if ‘m’ is a large enough number then it will take a long time to find the two prime factors. For example, in 1997 the estimated time required to factorize an ‘m’ about 150 digits long was about 200 years.

My example in the screenshots shows an ‘m’ which is 7 digits long. Such a number can be factorized in a fraction of a second using the [c_prime] custom function which forms part of this suite of programs—so my example is completely insecure! Double click on [TestPrime] and try it for yourself.

The Euler – Fermat Theorem

I must mention this theorem.

The Fermat of this Theorem is Pierre de Fermat (1601 – 1665) whose eponymous ‘Last Theorem’ has only recently been proved. Most of his discoveries were unrecognised (they were contained in personal letters) until Leonhard Euler published them a hundred years later. A hundred years before Euler this mathematical genius, Fermat proved what might be considered a simplified version of the EFT (Euler – Fermat Theorem).

Using the same notation as I have used this month, the EFT states that: (a^M) mod m = 1 (the ‘magic’ 1) for all ‘a’ which are relatively prime to the number ‘m’. In Fermat’s simplified version of this Theorem ‘m’ is prime and so M = (m − 1). Fermat’s simpler version a^(p − 1) mod p = 1 is often referred to as the FLT (Fermat’s Little Theorem).

Using the EFT I have been able to prove (to Colin Singleton’s satisfaction) that the RSA system ‘works’ provided that ‘d’ is the multiplicative inverse of ‘e’ modulus M. I have not been able to prove that the system works when ‘d’ is the multiplicative inverse of ‘e’ modulus L. This, coupled with the failure to mention the use of L in some revered texts makes me wonder if there are some circumstances in which the use of L is inappropriate (or just plain wrong); remember that my example uses L and not M.

If you have any light which you can cast on the merits or demerits of using L rather than M then please let me (and Colin) know.

A Puzzle

My example demonstrates the principles behind the RSA System of encryption. Using numbers which are relatively small (as I have done) is totally insecure and the code can be cracked using the custom functions I have included with this article.

The DHM system is much more secure even if relatively small numbers are used. How would you crack the DHM code? So that you can see what is involved, even with small numbers, here is an example for you to try to 'crack'. All the functions are available in the DHM directory. Find the shared value of K given that: g = 109, m = 256, E = 229, D = 85.

I know that, because of the very small numbers, the 'code' is vulnerable to a 'Trial and Error' attack looking for the 'e' which satisfies E = (g^e) mod m. At most there are 'm' of them, 256 in this case. What I am asking you to do is to find a method which does not rely on 'Trial and Error'. If you do use 'Trial and Error' then how many different values of 'e' satisfy the modulus equation 229 = (109^e) mod 256 ?

Back to Prime Numbers

I have always enjoyed Colin Singleton's Puzzle Corner so let me set one of my own.

What is the largest prime number which can be represented by the FPE?

This number represents a 'sort of' upper limit for those simple encryption methods in which numbers are represented in FPE format(s) rather than long character strings.

Finally

You can email or write to me at the addresses given in Paul's Fact File.

Please note that we no longer have a telephone or fax.