Number Theory and Cryptography (using Maple)

John Cosgrave

Department of Mathematics, St. Patrick's College, Drumcondra, Dublin 9, IRELAND

Abstract Since 1995-96 I have taught, using Maple, a yearly course on Number Theory and Cryptography to my undergraduate students[1]. In this paper I outline some basic number theoretical topics related to cryptography, based on my experience as a teacher of those topics. I am omitting all reference to practical teaching details, but will happily forward all teaching materials (notes, examination papers, etc.) to any interested readers. Finally, several of my NT and Cryptography course Maple worksheets[2] are available on the internet [Cos].

1 Introduction

My ideal reader of this paper is

- someone familiar with elementary number theory (essentially congruences, the Euclidean Algorithm, and Fermat's 'little' theorem), who would like to know how certain number theoretic ideas relate to the basic notions of Pohlig-Hellman (private-key) and Rivest-Shamir-Adleman public-key cryptography, or
- someone who knows some number theory, has never taught any cryptography, and who is wondering if it is something he/she might undertake.

Cryptography is the study of secure communication: how can two or more persons communicate securely with each other? The subject has a long and fascinating history, the best detailing of which is undoubtedly David Kahn's monumental The Codebreakers [K]. Also, it is well recognised that the two fundamental development in cryptography took place in the 1970's when W. Diffie and M.E. Hellman [DH] proposed the idea of public-key cryptography, and shortly afterwards R.L. Rivest, A. Shamir and L.M. Adleman [RSA] gave an actual realisation of the Diffie-Hellman proposal, the now classic

[1] Most of my students are training to be primary school teachers - 38 of them in my recent class - and have chosen Mathematics (by university requirement) as one of their 'academic' subjects.

[2] The best of which, if I may say so, entitled 'Bill Clinton, Bertie Ahern, and digital signatures', covers almost all the contents of this paper in an accessible (no theorems) manner.

RSA method. Even after the passage of some twenty years, the brilliance of those path-breaking papers has not diminished, and one can still profit from re-reading them.

The manner in which elementary number theory has made an impact on private and public-key cryptography is well known , and for my purposes may well be summarised as follows:

Given two parties[3] A and B who wish to communicate, A transforms her plaintext T (*Please send more money asap*) into numerical form N (a natural number formed as a result of some agreement, e.g. that 'a' is 1, 'b' is 2, etc.), and then, using some suitable 1-to-1 function f, computes $N' = f(N)$. A then communicates N' to B, who recovers N from N' using the inverse function f^{-1}, and then recovers A's original plaintext.

So far there is no Number Theory in any of this, and the above is merely an abstract mathematical formulation of the classic problem whose various solutions are beautifully and thrillingly described in Kahn's history [K].

1.1 Number Theory makes its entrance

How does Number Theory make its contribution to a solution of the classic problem of communication? It is all so startlingly simple, and may be summarised by saying that N' is formed by modular exponentiation with a specially chosen modulus, as is the recovery of N from N'. The essential idea may be conveyed with a simple, but unrealistic, example (realism merely involves better chosen larger moduli): suppose A wishes to send B the message consisting only of the single letter 'c,' the numerically transformed form of which is '3.' A could encrypt (disguise) '3' by forming $N' = f(3) = 3^7 (\mathrm{mod}\, 11)$ giving $N' = 9$ and then send '9' to B, who may then decrypt (recover) by forming $f^{-1}(N') \equiv 9^3 (\mathrm{mod}\, 11)$ which produces, for general reasons which will be clear in a moment, the original '3.'

Almost everything that one needs[4] is now easily explained: with the example just given, A could send any message, and have that message recovered by B, whose numerical value was in the range 1 to 10. That is a consequence of Fermat's 'little' theorem for the prime 11, as I now briefly illustrate.

Fermat's 'little' theorem [5] is the following result.

Theorem 1. *Let p prime, and let $a \in \mathbb{Z}$ with $a \not\equiv 0 (\mathrm{mod}\, p)$; then $a^{p-1} \equiv 1 (\mathrm{mod}\, p)$.*

Returning to our example we have, for any a in the range 1 to 10, that $a^{10} \equiv 1 (\mathrm{mod}\, 11)$, from which, by squaring both sides and multiplying both

[3] 'Alice and Bob.'

[4] For realistic Pohlig-Hellman (private-key) or Rivest-Shamir-Adleman (public-key) cryptography.

[5] One of the most remarkable elementary theorems, with a host of important consequences.

sides by a, we obtain $a^{21} \equiv a \,(\mathrm{mod}\, 11)$. Thus if A used general a in the range 1 to 10, then setting $N' = f(a) = a^7 (\mathrm{mod}\, 11)$ giving $N' = a'$, with a' chosen [6] in the range 1 to 10, and A then sends N' to B, who then decrypts by forming $f^{-1}(N') = (a')^3 \equiv (a^7)^3 \equiv a (\mathrm{mod}\, 11)$, returning the original a.

Theorem 1 underpins the Pohlig-Hellman cryptographic system, in much the same that the following Theorem 2 (what one might call the two prime version of the Euler-Fermat theorem) underpins the Rivest-Shamir-Adleman system. Texts dealing with Theorem 2 normally first prove the full version of the **Euler-Fermat theorem** (*Let n be a natural number, $n > 1$, and let a be any integer with $gcd(a, n) = 1$; then $a^{\phi(n)} = 1 (\mathrm{mod}\, n)$*, where $\phi(n)$ - the Euler phi-function - is the number of integers between 1 and $n - 1$ that are relatively prime to n), but such an appeal can be dispensed with, as seen in the following proof.

Theorem 2. *Let p and q be distinct primes, and let $a \in \mathbb{Z}$ with $a \not\equiv 0 (\mathrm{mod}\, p)$ and $a \not\equiv 0 (\mathrm{mod}\, q)$; then $a^{(p-1)(q-1)} \equiv 1 \,(\mathrm{mod}\, pq)$.*

Proof. Since $a \not\equiv 0 (\mathrm{mod}\, p)$ then $a^{p-1} \equiv 1 (\mathrm{mod}\, p)$, and $(a^{p-1})^{q-1} \equiv 1^{q-1} \equiv 1 (\mathrm{mod}\, p)$, and so $a^{(p-1)(q-1)} \equiv 1 (\mathrm{mod}\, p)$. Similarly $a^{(p-1)(q-1)} \equiv 1 (\mathrm{mod}\, q)$, and thus $a^{(p-1)(q-1)} \equiv 1 \,(\mathrm{mod}\, pq)$ since $\gcd(p,\ q) = 1$. □

2 Some technical number theory details and cryptographic applications

2.1 Relating the decryption power to the encryption power

Both the PH and RSA methods require computing the decryption power from the encryption power and the modulus, and for that one needs[7] the following result.

Theorem 3. *Let $m \in \mathbb{N}$ with $m > 1$, and let $e \in \mathbb{Z}$ with* $\gcd(e,\ m) = 1$; *then there is a unique $d \in \mathbb{Z}$ with $ed \equiv 1 (\mathrm{mod}\, m)$ and $1 \leq d \leq m - 1$.*

In applications

- m is $(p - 1)$, for prime p, in the PH system,
- m is $(p - 1)(q - 1)$, for distinct primes p and q, in the RSA system.

[6] This is an *important detail* in general: the modulus must have a value greater than the value of the numerical form of the plaintext. In the above unrealistic example where we used modulus 11, had A wished to send the message 'r' - whose numerical value would be 18 - then it would not be clear to B whether A was transmitting the letter 'r' or perhaps 'g,' whose numerical equivalent would be 7, and $18 \equiv 7 (\mathrm{mod}\, 11)$.

[7] Using the extended Euclidean algorithm.

2.2 Cryptographic applications: Pohlig-Hellman (private-key) and Rivest-Shamir-Adleman (public-key)

Fermat's little theorem, and the above two prime version of it then form the basis for the Pohlig-Hellman and Rivest-Shamir-Adleman cryptographic methods:

Theorem 4. *(The Pohlig-Hellman case.) Let*

- p *be prime, and* $e \in \mathbb{N}$ *with* $\gcd(e,\ p-1) = 1$,
- $P \in \mathbb{Z}$ *with* $P \not\equiv 0 (\mathrm{mod}\, p)$, *and* C *be defined by* $C \equiv P^e (\mathrm{mod}\, p)$, *and* d *be chosen so that* $d \in \mathbb{Z}$ *with* $ed \equiv 1 (\mathrm{mod}\,(p-1))$ *and* $1 \leq d < p-1$;

then [8] $C^d \equiv P (\mathrm{mod}\, p)$.

Proof. Since $C \equiv P^e (\mathrm{mod}\, p)$, then $C^d \equiv (P^e)^d \equiv P^{ed} (\mathrm{mod}\, p)$. (We need the $d \in \mathbb{N}$ to guarantee that $C^d \in \mathbb{Z}$.) Now, since $ed \equiv 1 (\mathrm{mod}\,(p-1))$ then $ed = m(p-1)+1$ for some $m \in \mathbb{Z}$, and, in fact, $m \in \mathbb{N}$ since $e,\ d \in \mathbb{N}$. Thus $C^d \equiv P^{ed} \equiv P^{m(p-1)+1} (\mathrm{mod}\, p)$, and by Fermat's 'little' theorem, we have $P^{p-1} \equiv 1 (\mathrm{mod}\, p)$. It follows that

$$C^d \equiv P^{m(p-1)+1} \equiv (P^{p-1})^m \times P \equiv 1^m \times P \equiv P (\mathrm{mod}\, p),$$

i.e. $C^d \equiv P (\mathrm{mod}\, p)$. □

Theorem 5. *(The Rivest-Shamir-Adleman case.) Let*

- p *and* q *be distinct primes, and* $e \in \mathbb{N}$ *with* $\gcd(e,\ (p-1)(q-1)) = 1$,
- $P \in \mathbb{Z}$ *with* $P \not\equiv 0 (\mathrm{mod}\, p)$, $P \not\equiv 0 (\mathrm{mod}\, q)$, *and* C *be defined by* $C \equiv P^e (\mathrm{mod}\, pq)$, *and*
- d *be chosen so that* $d \in \mathbb{Z}$ *with* $ed \equiv 1 (\mathrm{mod}\,(p-1)(q-1))$ *and* $1 \leq d < (p-1)(q-1)$;

then [9] $C^d \equiv P (\mathrm{mod}\, pq)$.

A proof may be given along the same lines as the earlier one.

[8] This now is the guarantee that when the numerical value of the 'plaintext' P is encrypted using 'e' - thus forming the numerical form of the 'ciphertext' C - and C is then decrypted using 'd', then the upshot of all of this is to recover P, the original plaintext (rather its numerical form, which one then translates back into ordinary text).

[9] Exactly the same comment here as in the previous footnote. The plaintext just gets jumbled up by the decryption power, and gets unscrambled by the decryption power. It's as simple as that!!

Encryption, decryption, and digital signatures First I set down the details for encryption and decryption for both the Pohlig-Hellman private-key, and the Rivest-Shamir-Adleman (RSA) public-key methods. The two methods are similar, but are also quite different: the private one is based on trust between the parties, whereas the private one is based on caution.

In both systems it is understood that the plaintext (the original, text form of the message "Please send me money as quickly as possible") is transformed into numerical form according to some agreed convention ('a' is 1, 'b' is 2, ... , 'z' is 26, 'A' is 27, 'B' is 28, etc. '0' is 53, '1' is 54, '2' is 55, etc.), and then that numerical form is itself transformed in some way. In both PH and RSA, that number (or blocks of numbers) is (are) subjected to modular exponentiation:

- where the modulus is a prime p, in the PH case,
- where the modulus n is the product of two primes in the RSA case.

The Pohlig-Hellman case How do two people, A and B, communicate using the PH private-key cryptographic method?

The details. Having shared their 'private keys', namely

- prime p (the modulus),
- encryption power e, and
- decryption power d,

related by

$$e \in \mathbb{N},\ \gcd(e,\ p-1) = 1, \qquad \text{and} \qquad ed \equiv 1 (\mathrm{mod}\, p-1),\ d \in \mathbb{N},$$

A and B proceed as follows: A (say) converts the plaintext T into numerical form P (say) by an agreed convention, and breaks that number P into numerical blocks

1. $P_1, P_2, \ldots, P_r$, each having positive numerical value less than p.
2. A then forms the numbers $C_1, C_2, \ldots, C_r$ ('C' for cipher) as follows (this is the **encryption** of the P's):

$$C_1 \equiv P_1^e (\mathrm{mod}\, p),\ C_2 \equiv P_2^e (\mathrm{mod}\, p), \ldots, C_r \equiv P_r^e (\mathrm{mod}\, p),$$

where the values of $C_1,\ C_2,\ \ldots\ , C_r$ are chosen so their numerical values are positive and less than p. A then transmits those numerical blocks $C_1,\ C_2,\ \ldots\ , C_r$ to B.

On receipt of those blocks of numbers, B proceeds to decrypt (and so recover the original plaintext) by computing the numbers $c_1,\ c_2,\ \ldots\ , c_r$:

$$c_1 \equiv C_1^d (\mathrm{mod}\, p),\ c_2 \equiv C_2^d (\mathrm{mod}\, p),\ \ldots\ , c_r \equiv C_r^d (\mathrm{mod}\, p),$$

with the values of $c_1,\ c_2,\ \ldots\ , c_r$ chosen so that they are positive and less than p.

Then - and this is now the whole point of all of this - those numbers $c_1,\ c_2,\ \ldots\ , c_r$ are, in fact, the numbers $P_1,\ P_2,\ \ldots\ , P_r$.

The Rivest-Shamir-Adleman case How do two people, A and B, communicate using the RSA public-key cryptographic method?

The details. A (say) having chosen his/her 'keys'namely

- $n = p \times q$ (n is the modulus), for distinct (and, in practice, large [10]) primes p and q,
- encryption power e, and
- decryption power d

related by

$$e \in \mathbb{N},\ \gcd(e,\ (p-1)(q-1)) = 1, \quad \text{and} \quad ed \equiv 1 (\mathrm{mod}\,(p-1)(q-1)),\ d \in \mathbb{N}.$$

A and B proceed as follows: A (say) having made his/her 'public-key', namely $(n,\ e)$, known to B, B would then communicate with A as follows: B would convert the plaintext T into numerical form P (say) by an agreed convention, and would break that number P into numerical blocks:

1. $P_1,\ P_2,\ \ldots\ ,\ P_r$, each having positive numerical value less than n.
2. B then forms the numbers $C_1,\ C_2,\ \ldots\ ,\ C_r$ ('C' for cipher) as follows (this is the encryption of the P's):

$$C_1 \equiv P_1^e (\mathrm{mod}\, n),\ C_2 \equiv P_2^e (\mathrm{mod}\, n),\ \ldots\ ,\ C_r \equiv P_r^e (\mathrm{mod}\, n),$$

where the values of $C_1,\ C_2,\ \ldots\ ,\ C_r$ are chosen so they are positive and less than n. B then transmits those numerical blocks $C_1,\ C_2,\ \ldots\ ,\ C_r$ to A.

On receipt of those blocks of numbers, A proceeds to decrypt by computing the numbers $c_1,\ c_2,\ \ldots\ ,\ c_r$:

$$c_1 \equiv C_1^d (\mathrm{mod}\, n),\ c_2 \equiv C_2^d (\mathrm{mod}\, n),\ \ldots\ ,\ c_r \equiv C_r^d (\mathrm{mod}\, n),$$

with the values of $c_1,\ c_2,\ \ldots\ ,\ c_r$ chosen so that they are positive and less than n. Then - and again this is now the whole point of all of this - those numbers $c_1,\ c_2,\ \ldots\ ,\ c_r$ are, in fact, the numbers $P_1,\ P_2,\ \ldots\ ,\ P_r$.

2.3 'Signing' messages using the RSA cryptographic method.

Suppose you received a message from someone; how would you know the message really came from them? For example, suppose you received the following message: *Please call to see me on Wednesday at 3.00 P.M. John Cosgrave.*

It would be almost certain that the message came from me, especially if I signed it, and you know what my signature looks like. However, someone could have forged my signature, and you would be misled into thinking that

[10] But, and again in practice, not just 'large', but one would have to be careful about choosing the p and q so that n could not be easily factored.

I had asked you to visit me. You would turn up at my office on Wednesday at 3.00 P.M., and (possibly) find that I wasn't there ... . Of course it wouldn't really matter; the worst that would have happened is that you would have wasted your time.

Suppose, though, that an army general received a message saying something like: *At 6.00 A.M. tomorrow, send 1,000 troops to Place X ... (Signed by the) Commander-in-Chief.* How can the general be certain that the message really has come from the C.-in-C.?

In earlier times, documents or messages were authenticated by a physical signature or seal (though they could have been forged). In recent times there is increasing reliance on electronic means of communication (by government, diplomatic circles, military, business, banking, political groupings, criminal organisations, private individuals, etc.) which do not allow, of course, for a physical signature. With electronic communication, authentication is guaranteed by a 'digital signature,' and this is how it is done:

Recall the connection between e and d namely $ed \equiv 1 (\mathrm{mod}\, (p-1)(q-1))$ and note that it can be rewritten as $de \equiv 1 (\mathrm{mod}\, (p-1)(q-1))$, where the e and d have simply been interchanged. That simple interchanging has a very, very powerful consequence: it enables a user of RSA to sign a message. This is all they have to do:

To illustrate, let us return to my earlier: "For example, suppose you received a message saying: *Please call to see me on Wednesday at 3.00 P.M. John Cosgrave.*" This is what I can do (assuming I am a user of RSA, and you know my public-key (n, e)) that will convince you that the note you receive from me, really is from me:

I can encrypt a message to you by:

- using my private decryption power d (which only I know) as my encryption power.

Then, on receipt of my message, you can decrypt it by:

- using my public encryption power d (which you, and possibly others, know) as your decryption power.

Anticipating an objection. You might (rightly) say that anyone who can intercept my message, and who knows my public-key, can also decrypt my message to you. That is a simple fact (which is best illustrated in a Maple worksheet).

Fortunately public-key cryptography once again comes to our rescue. If I want to 'sign' my message to you and I don't want anyone but you to be able to read the contents of my message to you, I can then achieve my aim by performing a double encryption [11] :

[11] Assuming you are using RSA, and I know your public-key.

- First [12] I use my private-key to do an initial encryption (and in the process 'sign' my message to you),
- then I use your public-key to perform a second encryption before sending my message.

When you receive my doubly-encrypted message you can then read it by performing a double decryption:

- First you use your private-key to perform the first decryption,
- then you use my public-key to perform the second decryption, and so read my message.

Remark 6. A way of visualising this. Think of public-key cryptography in terms of paints and paint-removers. My public-key is some paint which I have made, and which, if it is used, only I can remove by applying the secret paint remover which I also have made, and which only I have access to.

You have to allow your imagination to let the paint and paint remover to be used in reverse!! By that I mean that if something is covered with my paint then not only can it be uncovered by applying my paint remover, but that the same is true in reverse: if something is first covered with my paint remover then what is now there can be uncovered by applying my paint!!

Anyone who wishes to send me a secret message simply writes a message, gets some of my paint, and paints (encryption) over my message. When I receive your message I apply my paint remover (decryption) to it, and so read your message. Everything I have said about 'I' applies to you: you have your paint and your paint remover ... .

Now form a mental image of what I have described above:

- First I use my secret paint remover to do an initial painting (encryption, and in the process 'sign' my message to you),

- then I use your public paint to perform a second encryption before sending my message.

When you receive my doubly-encrypted message you can then read it by performing a double decryption:

- First you use your private paint remover to perform the first decryption,
- then you use my public paint to perform the second decryption, and so read my message.

[12] Actually which I do first depends on whether my public modulus is smaller than your public modulus:

- If my public modulus is smaller than yours then what I have described above is in fact what I would (and should) do, but:
- If my public modulus is greater than yours then what I have described above should be done in reverse order, by me and you.

Basic understanding: We assume that A has public key $(n_A,\ e_A)$ with private key d_A and that B has public key $(n_B,\ e_B)$ with private key d_B.

Question: How can B 'sign' a message to A (equally A send one to B) so that A can have confidence that the message received has come from B?

Answer: It can be done quite easily, but it depends on which is the smaller: n_A or n_B. That is, it depends on whether:

(i) $n_B < n_A$, or
(ii) $n_A < n_B$.

The details: Let us suppose that (i) happens [13] (namely that $n_B < n_A$), and suppose that B wants to (securely) send and 'sign' a message P to A.

We make the usual understanding that P (the numerical form of the plaintext) has been put into numerical form according to some convention (a is 1, b is 2, etc.).

This, then, is what B does to (securely) send and 'sign' a message P to A.

1. B breaks P up into blocks of numbers. $P_1,\ P_2,\ \ldots\ ,\ P_r$, each having numerical value less than n_B (and so are *automatically* less than n_A since $n_B < n_A$), and each relatively prime to both n_B and n_A. B then 'signs' using his/her private key by doing this:
2. B forms the numbers $c_1,\ c_2,\ \ldots\ ,\ c_r$ as follows:

$$c_1 \equiv P_1^{d_B} (\mathrm{mod}\, n_B),\ c_2 \equiv P_2^{d_B} (\mathrm{mod}\, n_B), \ldots, c_r \equiv P_r^{d_B} (\mathrm{mod}\, n_B),$$

   the $c_1,\ c_2,\ \ldots\ ,\ c_r$ chosen with positive values, less than n_B.
3. B then forms the following blocks of ciphertext, and sends those to A:

$$C_1 \equiv c_1^{e_A} (\mathrm{mod}\, n_A), C_2 \equiv c_2^{e_A} (\mathrm{mod}\, n_A), \ldots, C_r \equiv c_r^{e_A} (\mathrm{mod}\, n_A),$$

   the $C_1,\ C_2,\ \ldots\ ,\ C_r$ chosen with positive values, less than n_A.

In summary,

1. B first signs with their own private key,
2. and then sends the newly formed ciphertexts in the usual way, using A's public key.

On receipt of the $C_1,\ C_2,\ \ldots\ ,\ C_r$ this is what A does:

1. A partially decrypts the numbers $C_1,\ C_2,\ \ldots\ ,\ C_r$ using their own private key, by forming the numbers $x_1,\ x_2,\ \ldots\ ,\ x_r$ as follows:

$$x_1 \equiv C_1^{d_A} (\mathrm{mod}\, n_A), x_2 \equiv C_2^{d_A} (\mathrm{mod}\, n_A), \ldots, x_r \equiv C_r^{d_A} (\mathrm{mod}\, n_A),$$

   the $x_1,\ x_2,\ \ldots\ ,\ x_r$ chosen with positive values, less than n_A. (Those $x_1,\ x_2,\ \ldots\ ,\ x_r$ are, of course, none other than $c_1,\ c_2,\ \ldots\ ,\ c_r$.)

[13] Later we will see what to do if (ii) happens.

2. A completes the decoding by forming the following blocks of ciphertext:

$$y_1 \equiv x_1^{e_B} (\mathrm{mod}\, n_B), y_2 \equiv x_2^{e_B} (\mathrm{mod}\, n_B), \ldots, y_r \equiv x_r^{e_B} (\mathrm{mod}\, n_B),$$

the $y_1,\ y_2,\ \ldots\ ,\ y_r$ chosen with positive values, less than n_B.

The whole point is now that those $y_1,\ y_2,\ \ldots\ ,\ y_r$ are none other than the numerical form of B's original plaintext message, namely $P_1,\ P_2,\ \ldots\ ,\ P_r$.

If, however we had $n_A < n_B$, then this is what B would do [14] : suppose that B wants to send message P to A. B breaks P up into blocks of numbers $P_1,\ P_2,\ \ldots\ ,\ P_r$ (each having value less than n_A (and so are *automatically* less than n_B since $n_A < n_B$), and each relatively prime to both n_A and n_B). This, then, is what B does to 'sign' the message to A:

First B forms the numbers $c_1,\ c_2,\ \ldots\ ,\ c_r$ by using A's public key, just as in an ordinary unsigned message, as follows:

$$c_1 \equiv P_1^{e_A} (\mathrm{mod}\, n_A), c_2 \equiv P_2^{e_A} (\mathrm{mod}\, n_A), \ldots, c_r \equiv P_r^{e_A} (\mathrm{mod}\, n_A),$$

the $c_1,\ c_2,\ \ldots\ ,\ c_r$ chosen with positive values, less than n_A.

Then B (and this is what 'signs' for B, the using of B's 'secret') sends to A the following blocks of ciphertext:

$$C_1 \equiv c_1^{d_B} (\mathrm{mod}\, n_B), C_2 \equiv c_2^{d_B} (\mathrm{mod}\, n_B), \ldots, C_r \equiv c_r^{d_B} (\mathrm{mod}\, n_B),$$

the $C_1,\ C_2,\ \ldots\ ,\ C_r$ chosen with positive values, less than n_B. In short, B first encodes in the usual way using A's public key, and then 'signs' using their own private key.

On receipt of $C_1,\ C_2,\ \ldots\ ,\ C_r$ this is what A does:

First A partially decodes the numbers $C_1,\ C_2,\ \ldots\ ,\ C_r$ using B's public key, by forming the numbers $x_1,\ x_2,\ \ldots\ ,\ x_r$ as follows:

$$x_1 \equiv C_1^{e_B} (\mathrm{mod}\, n_B), x_2 \equiv C_2^{e_B} (\mathrm{mod}\, n_B), \ldots, x_r \equiv C_r^{e_B} (\mathrm{mod}\, n_B),$$

the $x_1,\ x_2,\ \ldots\ ,\ x_r$ chosen with positive values, less than n_B. (Those $x_1,\ x_2,\ \ldots\ ,\ x_r$ are, of course, none other than $c_1,\ c_2,\ \ldots\ ,\ c_r$.

Then A completes the decoding by forming the following blocks of ciphertext:

$$y_1 \equiv x_1^{d_A} (\mathrm{mod}\, n_A), y_2 \equiv x_2^{d_A} (\mathrm{mod}\, n_A), \ldots, y_r \equiv x_r^{d_A} (\mathrm{mod}\, n_A),$$

the $y_1,\ y_2,\ \ldots\ ,\ y_r$ chosen with positive values, less than n_A. The whole point is, again, that those $y_1,\ y_2,\ \ldots\ ,\ y_r$ are none other than the numerical form of B's original plaintext message, namely $P_1,\ P_2,\ \ldots\ ,\ P_r$.

All of this is best illustrated in a Maple worksheet, and such details, actually carried out, may be seen in my 'Clinton ... ' Maple public lecture [Cos].

[14] Actually all that B and A do is to do what they previously did, except to do it in *reverse order*.

3 Some elementary, but non-trivial primality testing methods

Is the converse of Fermat's little theorem true? That is, if $n \in \mathbb{N}(n \geq 2)$, and $a \in \mathbb{Z}$ such that $a^{n-1} \equiv 1(\mathrm{mod}\, n)$, is n necessarily prime? It is well known that it isn't, as the example [15] $2^{340} \equiv 1(\mathrm{mod}\, 341)$ shows [16].

Lucas (starting in 1876) observed the first of a series of partial converses to Fermat's 'little' theorem. These results had the following common form: if $n \in \mathbb{N}(n \geq 2)$, and $a \in \mathbb{Z}$ such that $a^{n-1} \equiv 1(\mathrm{mod}\, n)$, then - *providing some extra condition is satisfied* - n is prime.

The start of a serious study of primality testing I will restrict myself to the methods of Lucas (1876-78), Proth (1878), Pocklington (1914), Lehmer (1927) and Selfridge (1967), and I begin with the following result.

Theorem 7. *(Lucas, 1876). Let* $n \in \mathbb{N}(n \geq 3)$, *and suppose there is an* $a \in \mathbb{Z}$ *such that* $a^{n-1} \equiv 1(\mathrm{mod}\, n)$ *and* $a^x \not\equiv 1(\mathrm{mod}\, n)$ *for all* x *with* $1 \leq x < n-1$; *then* n *is prime.*

Alternative wording of this theorem. *Let* $n \in \mathbb{N}(n \geq 3)$, *and suppose there is some* $a \in \mathbb{Z}$ *such that* $ord_n a = n-1$*; then* n *is prime.*

This theorem which appears at first sight to be so weak (but which ultimately isn't, in the sense that it can be gradually improved bit by bit to produce wonderfully effective results) marked the start of modern primality testing. As a test it is even worse than the Eratosthenes method, but Lucas himself improved upon it in 1878 by showing that the condition " $a^x \not\equiv 1(\mathrm{mod}\, n)$ for all x with $1 \leq x < (n-1)$ " could be replaced with the less restrictive one that "$a^x \not\equiv 1(\mathrm{mod}\, n)$ for all x *with* $1 \leq x < (n-1)$ with $x \,|\, n-1$." However, even that improvement ceases to be useful whenever $n-1$ has a lot of factors.

Example 8. A Maple computation which conveys the idea of a proof of Lucas' 1876 theorem. Here I use Maple to compute all powers of 2 modulo 101 from the 1st to the 100th power:

```
>seq(2&^x mod 101, x=1..100);  # here 'a' is 2, and 'n' is 101
```

2, 4, 8, 16, 32, 64, 27, 54, 7, 14, 28, 56, 11, 22, 44, 88, 75, 49, 98, 95, 89, 77, 53, 5, 10, 20, 40, 80, 59, 17, 34, 68, 35, 70, 39, 78, 55, 9, 18, 36, 72, 43, 86, 71, 41, 82, 63, 25, 50, 100, 99, 97, 93, 85, 69, 37, 74, 47, 94, 87, 73, 45, 90, 79, 57, 13, 26, 52, 3, 6, 12, 24, 48, 96, 91, 81, 61, 21, 42, 84, 67, 33, 66, 31, 62, 23, 46, 92, 83, 65, 29, 58, 15, 30, 60, 19, 38, 76, 51, 1

[15] First noted by Sarrus in 1814.

[16] 341 is an example of a **pseudoprime** to the base 2; that is it is a composite n satisfying $2^{n-1} \equiv 1(\mathrm{mod}\, n)$.

Noting that $2^{100} \equiv 1 (\mathrm{mod}\, 101)$ and $2^x \not\equiv 1 (\mathrm{mod}\, 101)$ for all x with $1 \leq x < 100$, one should make a critical observation, namely: there are 100 outputs, and *no two of those outputs are equal*, and that, as a *consequence*, all residues between 1 and 100 must occur (exactly once), and that as a further *consequence* 101 must be prime. Why? Well, if 101 were composite then it would have as a factor some prime smaller than 101. Let's suppose it had 7 (say) as a factor, then $2^x \equiv 7 (\mathrm{mod}\, 101)$ for some x, would imply 2 is divisible by 7.

Several similar examples now reduce the proof of Lucas' theorem to a formality.

Proof. (of Lucas' 1876 theorem) [17] Suppose n is composite. We will show that is impossible, and so n must be prime. We show that $a^1,\ a^2, \ldots,\ a^{n-1}$ are congruent mod n, in some order, to $1,\ 2, \ldots,\ n-1$, and then argue that is impossible.

None of $a^1,\ a^2, \ldots,\ a^{n-1}$ is 0 mod n, because if $a^m \equiv 0 (\mathrm{mod}\, n)$ for some $m \in \mathbb{N}$, then $a^m = nX$, for some $X \in \mathbb{Z}$. Now, let p be a prime with $p|n$; we would have $p|a^m$, and so would have $p|a$. But $p|n$ and $p|a$ would conflict with $\gcd(a,\ n) = 1$, and so none of $a^1,\ a^2, \ldots,\ a^{n-1}$ is 0 mod n.

Also, if $a^v \equiv a^u (\mathrm{mod}\, n)$ for some $u,\ v,\ 1 \leq u < v \leq n-1$, then $a^u(a^{v-u} - 1) \equiv 0 (\mathrm{mod}\, n)$, and from $\gcd(a^u,\ n) = 1$, it follows that $(a^{v-u} - 1) \equiv 0 (\mathrm{mod}\, n)$. Setting $x = v - u$, we have $a^x \equiv 1 (\mathrm{mod}\, n)$, where $1 \leq x \leq (n-2)$. That conflicts with the second condition of Lucas' theorem, and it follows that no two of $a,\ a^2, \ldots,\ a^{n-1}$ are congruent to each other mod n.

Thus $a,\ a^2, \ldots,\ a^{n-1}$ are congruent mod n, in some order, to $1,\ 2, \ldots,\ n-1$, and so for some integer r $(1 \leq r \leq n-2)$ we have $a^r \equiv p (\mathrm{mod}\, n)$, where p is the earlier prime dividing n (and so $1 < p < n$). That is impossible since $a^r \equiv p (\mathrm{mod}\, n)$ means $a^r \equiv nX' + p$, for some $X' \in \mathbb{Z}$, from which, with $p|n$ we obtain $p|a^r$. We would have $p|a$. That conflicts with $\gcd(a,\ n) = 1$, and so n cannot be composite. Thus n is prime. □

Theorem 9. *(what I call the 'Lucas-Kraitchik-Lehmer* [18] *Theorem'). Let* $n \in \mathbb{N}$ *with* $n > 1$, *and suppose there is some* $a \in \mathbb{Z}$ *with* $a^{n-1} \equiv 1 (\mathrm{mod}\, n)$ *and* $a^{\frac{n-1}{p}} \not\equiv 1 (\mathrm{mod}\, n)$ *for all primes* p *with* $p|(n-1)$*; then* n *is prime.*

[17] The original Lucas proof involved using a theorem (the famous 'Euler-Fermat theorem') whose own proof involves quite a lot of extra work. Here I give a proof which avoids such a reference.

[18] It is, of course, Lehmer's theorem of 1927. In my first couple of years of teaching a proof of this I used Lehmer's original proof, but, as anyone familiar with that proof will know, a very heavy and quite unnecessary use is made of the Euler ϕ-function, and my students had great difficulty in following it. Fortunately the proof I subsequently gave, here in this paper, was more readily understood (when properly motivated).

Remark 10. This is a very powerful theorem, whose power is only properly realised when Maple, or other similar work is performed with very large numbers.

Proof. (the strategy of the proof is to show that $ord_n a = n-1$, and it follows from Lucas' theorem of 1876 that n is prime.) Let $r = ord_n a$, then $n-1 = rR$, some $R \in \mathbb{N}$. If $R > 1$, then $R = pR'$, some prime p, and $R' \in \mathbb{N}$. Thus $n-1 = rR = rpR'$, $\frac{n-1}{p} = rR' \in \mathbb{N}$, so p divides $(n-1)$. Then $(a^r)^{R'} \equiv 1^{R'} \equiv 1 (\mathrm{mod}\, n)$, and thus $a^{\frac{n-1}{p}} = a^{rR'} \equiv 1 (\mathrm{mod}\, n)$ which conflicts with the second condition of the theorem. Thus $R \not> 1$, and $r = ord_n a = n-1$. By Lucas' 1876 theorem it follows that n is prime. □

Remark 11. The Lehmer 1927 theorem is sometimes referred to, for obvious reasons, as the '$\frac{n-1}{p}$' theorem. There is a further important improvement (dating from 1967) of D. H. Lehmer's theorem that is due to another U.S. mathematician John Selfridge. One only appreciates the value of Selfridge's improvement after one has had experience with using the Lehmer theorem with Maple computations. Selfridge's theorem is sometimes referred to as the 'change of base' theorem, for reasons which will become apparent when one uses it.

Theorem 12. *(what I call the 'Lucas-(Kraitchik)-Lehmer-Selfridge Theorem') Let $n \in \mathbb{N}$ with $n > 1$, and suppose that for each prime p_i with $p_i | (n-1)$ there is some $a_i \in \mathbb{Z}$ with $a_i^{n-1} \equiv 1 (\mathrm{mod}\, n)$ and $a_i^{\frac{n-1}{p_i}} \not\equiv 1 (\mathrm{mod}\, n)$; then n is prime.*

Another important variation is Pocklington's theorem.

Theorem 13. *(Pocklington, 1914) Let $n-1 = UF = Up_1^{\alpha_1} p_2^{\alpha_2} \ldots p_r^{\alpha_r}$ be an incomplete factorisation of $n-1$ (where U is the 'unfactored part' of $n-1$, and $F = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_r^{\alpha_r}$ is its factored part) with $U < F$ and $\gcd(U, F) = 1$. Suppose there is an a such that $a^{n-1} \equiv 1 (\mathrm{mod}\, n)$ and $\gcd(a^{\frac{n-1}{p_i}} - 1,\ n) = 1$ for all i, $1 \leq i \leq r$; then n is prime.*

Example 14. Let p_r denote the r*th* prime.

- I have used LKLS to prove the primality of the 1006-digit $2^{50}(p_1 p_2 \ldots p_{20})^3 + 1$, and
- the primality of the 1405-digit $2^{37} 1!2!3!4! \ldots 48!49!50! + 1$.
- Also I have used Pocklington to establish the primality of the (serendipitously found) 2000-digit $p_1 p_2 \ldots p_{325} p_{326}^{325} + 1$ (see [Cos] for the Maple worksheet details), and also
- the primality of the 3318-digit $p_1 p_2 \ldots p_{346} p_{347}^{346} p_{348}^{346} + 1$.

Another interesting elementary result is Proth's (1878), the standard version of which is the following result.

Theorem 15. *Let* $N = s \cdot 2^r + 1$, *where* s, $r \in \mathbb{N}$ *and*[19] $s < 2^r$. *Suppose there is an* $a \in \mathbb{Z}$ *such that* $a^{\frac{N-1}{2}} \equiv -1 (\mathrm{mod}\, N)$; *then* N *is prime.*

I have made the minor improvement of the $s < 2^r$ condition to $s \leq 2^r + 1$, with this proof.

Proof. First, note the standard result [20] about prime divisors of Fermat type numbers: let $x \in \mathbb{Z}$ and $m \in \mathbb{N}$, then every odd prime divisor q of $x^{2^m} + 1$ satisfies $q \equiv 1 (\mathrm{mod}\, 2^{m+1})$. For p a prime divisor of N, we have $a^{\frac{N-1}{2}} \equiv (a^s)^{2^{r-1}} \equiv -1$ and so $p \equiv 1 (\mathrm{mod}\, 2^r)$. If N is composite, then N is a product of at least two primes each of which has minimum value $2^r + 1$, and so $N = s \times 2^r + 1 \geq (2^r + 1)(2^r + 1) = 2^r \times 2^r + 2 \times 2^r + 1$. It follows that $s \geq 2^r + 2$, which is incompatible with $s \leq 2^r + 1$. Thus N is prime. □

4 Some elementary, but non-trivial factorisation methods

Maple has a number of factorisation commands, the default one of which [21] is the 1975 continued fraction method of Morrison and Brillhart. It also has the command `ifactor(n, pollard)` which puts into effect the Pollard $\rho-$method with the Floyd cycle algorithm improvement, but only using iterates of '2' using the function $x^2 + 1$.

I am keen that my students should have exposure to some non-trivial factorisation methods, and have narrowed myself down to just two [22]:

- Pollard's $p - 1$ method (1974), which uses Fermat's 'little' theorem,
- Pollard's $\rho-$method (1974), which uses a generalisation of the *birthday paradox.*

It is my experience that students are really fascinated by both Pollard methods, and I can assure any reader that the inclusion of these methods in such a course is a source of very great excitement in the classroom. Personally I never really appreciated these methods until I decided to teach them, and they form one of the highlights of the course. Many students are greatly

[19] Some texts add an entirely irrelevant requirement that s be *odd.*

[20] Which I prove for my students, using standard order theorems and Fermat's little theorem.

[21] The Maple command is ifactor(n).

[22] With as much details as possible. I also expose them to the elementary Fermat method, which, although it only requires high school mathematics to understand it, is nevertheless one which can't be ignored in choosing two primes for RSA usage. Many students are greatly impressed by seeing the product of two really large primes - with hundreds of digits, but which differ by only several thousands - being factored almost instantly by the Fermat method. It allows one to drive home the point that mere size is not enough when choosing primes for RSA usage.

impressed with how effective both methods are, especially the $p-1$ method when used on RSA type numbers where one of the primes has been formed by using a Lehmer-Selfridge type construction. I refer my reader to the example in my Maple public lecture 'Bill Clinton, ... ' [Cos].

My approach to teaching the Pollard methods In teaching my students the Pollard methods I abandon all reticence, and try to impress on them that in studying these methods they are considering the work of a master mathematician with an extraordinary, fertile imagination. The first point I make is that these two methods, which appear so different at first, are in fact driven by a single, apparently useless, but actually incredibly powerful, common idea. Given a composite n, known to be composite because of failing a Lucas-Fermat test to some base [23] the common idea in the methods is to attempt to find some integer M (I urge my students that they think of 'M' as being short for Magic, because in the two Pollard methods it really is magical the manner in which he creates this M) such that $\gcd(M,\ n) > 1$ and $\gcd(M,\ n) < n$.

Since Pollard's approach can - and indeed does - appear very, very strange to weaker students, then I play on that perception, and indeed I attempt to rubbish the idea before I even show them how very powerful it is. While pointing out that finding such an M would of course mean that one had found a proper factor of n, I do concede that the idea could appear completely useless because of these considerations:

- How is one going to find such an M? ...
- By trial and error? Let's see if $M = 2,\ 3,\ 5,\ 7, \ldots$ would do? Why, that would be even worse (because of the gcd computation) than using the Eratosthenes approach of trying possible factors up to $\sqrt{n}$.

However I then put it to them that they should consider it a tribute to Pollard's fertile imagination that he was able to conceive two wonderful realisations [24] of finding this elusive M. These methods will be known to my reader, and so I will only briefly describe how I attempt to convey Pollard's thinking to my students. For both methods I emphasise that Pollard's hope is to find one of the proper factors p of n (and not necessarily the least one!).

In the case of his 1974 method, his $(p-1)$ method, he attempts to find that 'p' by exploiting Fermat's little theorem in the following way.

Lemma 16. *For p any prime and $a \in \mathbb{Z}$, $a \not\equiv 0 (\mathrm{mod}\, p)$, one has $a^{k!} \equiv 1 (\mathrm{mod}\, p)$, and thus $a^{k!} - 1 \equiv 0 (\mathrm{mod}\, p)$, for all sufficiently large values of k.*

[23] Normally (but not always) that $2^{n-1} \not\equiv 1 \,(\mathrm{mod}\, n)$.

[24] It is a well known joke amongst mathematicians that a trick is something that works once, while an idea is something that works twice (or more). I have often wondered if Pollard had an idea in 1974, or simply a trick.

(That '$(a^{k!}-1)$', hopefully with a not too large value for k, is going to be ones 'M': it is divisible by p, as is n, and ones ambition is to quickly find a reasonably small k, so that the gcd of n and $(a^{k!}-1)$ comes to be greater than 1, but less than n.) That result is, of course, a trivial consequence of Fermat's little theorem, since it is certainly true for $k \geq p-1$.

Proof. From Fermat's 'little' theorem we have $a^{p-1} \equiv 1 (\mathrm{mod}\, p)$. Also, for sufficiently large k we have $(p-1)|k!$ (e.g., $k \geq p-1$ would do trivially), and so $k! = (p-1)K$, for some $K \in \mathbb{N}$. Then $(a^{p-1})^K \equiv 1^K \equiv 1 (\mathrm{mod}\, p)$, and so $a^{(p-1)K} \equiv 1 (\mathrm{mod}\, p)$, *i.e.* $a^{k!} \equiv 1 (\mathrm{mod}\, p)$. □

Pollard's insight was that although the latter congruence is trivial, nevertheless because of the way in which the prime factorisation structure of $k!$ behaves as k increases in size, one may have $a^{k!} \equiv 1 (\mathrm{mod}\, p)$ for substantially smaller values of k than the trivial $k = p-1$.

Example 17. For example, if $p = 97$, then the trivial $a^{96!} \equiv 1 (\mathrm{mod}\, 97)$ may be vastly improved upon with $a^{8!} \equiv 1 (\mathrm{mod}\, p)$. That is, because $97-1 = 96 = 2^5 \times 3^1$, and $8! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 = 2^1 \times 3^1 \times 2^2 \times 5 \times (2^1 \times 3^1) \times 7 \times 2^3 = 2^6 \times 3^2 \times 5 \times 7$, is divisible by 96, because of the appropriate accumulation of 2's and 3's in the prime decomposition of 8!.

Thus if one had a composite n (reasonably big, say) which, unknown to one, happened to have 97 as a factor, then that fact would be quickly revealed by successively calculating the early terms of the sequence

$$\gcd(2^{1!}-1,\ n), \gcd(2^{2!}-1,\ n), \gcd(2^{3!}-1,\ n), \ldots .$$

The real work that has to be done to get the idea across may be seen in much greater detail in my related Maple worksheet [Cos]. Suffice it here to say that the key Maple programming computational steps are not to compute each $\gcd(a^{k!}-1,\ n)$ from scratch, but rather to do them recursively, and furthermore not to compute actual values of the $a^{k!}-1$, but rather their reduced values $\mathrm{mod}\, n$.

This, then, is the final [25] version of the Maple procedure I lead my students to the following algorithm.

```
> Pollard := proc(seed, n)
  local a, k; a[1] := seed:
  for k from 2 while igcd(n, a[k-1]-1 mod n)=1
  do a[k] := a[k-1]&^k mod n od;
  if igcd(n, a[k-1]-1 mod n) < n then
  lprint(‘After‘, k-1, ‘steps we find that‘,
```

[25] With my students I deliberately build up through slower stages to make certain points, as will be seen by anyone who reads my detailed Maple worksheet. I could, of course, dispense with the '`lprint`' line, and simply output a proper factor if one is found, otherwise have no output.

```
  igcd(n, a[k-1]-1 mod n),‘is a proper factor of‘, n)
  else lprint(‘No proper factor found; try some other seed‘
  fi end:
```

It is my experience that most students are greatly impressed with the effectiveness of this Pollard inspired, Maple procedure. Cryptographic examples of it in action may be read in details given in my Maple public lecture ‘Bill Clinton, Bertie Ahern, and digital signatures’ [Cos].

For example, if one performed the following Maple computations: first, create two large primes p and q, the first of which entails $p-1$ having only small prime divisors, and then formed their product n, one could verify that n is composite by showing it fails a base 2 Lucas-Fermat test, and also factor n using the above Pollard procedure.

These computations are all quickly executed:

```
>p := 2^40*3^52*7^52  1;+ #an 81-digit prime
```

p := 626041542350318657267231514757451147212314337872408353488806911360306887054170521 7

```
>q := nextprime(10^66  12345678910987654321) +# 67 digits
```

q := 10000000000000000000000000000000000000000000000123456789109876544 47

```
>n := p*q: # value suppressed. 147 digits
>2&^(n-1) mod n; # shows ‘n’ is not prime:
```

315212795823765344257786197018813662048038750896292204273838589012443276119009530179154090994865246547633014859071848668063492151077465700747069971.

Finally, execute the following Maple command, which takes only seconds:

```
>Pollard(2, n); # choosing ‘seed’ to be ‘2.’
```

Output:

After 322 steps we find that
626041542350318657267231514757451147212314337872408353488069113603068870541705217
is a proper factor of
626041542350318657267231514757451147212314337880137261354865627195181479941250685466423117165291499989527024421354240887817669535791655926533149999.

4.1 Some brief comments on the Pollard $\rho-$method

Suffice it to say that Pollard suggested another remarkable way of arriving at an M' with the desirable properties listed earlier, except that M is now not arrived at as a consequence of constructing a sequence one of whose eventual terms is the desired M, but rather - as a consequence of the 'generalised birthday paradox' - so that M is arrived at by forming differences. This method is very well explained in Pomerance's MAA notes [P1], or in Koblitz's book [Ko], and I refer my reader to those sources. In treating this method with my own students I explain the original Pollard approach, eventually arrive at the classic $\rho-$figure, and discuss how the computation may be speeded up by using the Floyd cycle finding algorithm.

Any reader already familiar with Pollard's $\rho-$method will know that Pollard himself suggested starting with seed '2' and using iterated values of $x^2 + 1 \pmod{n}$ as the means of producing the random sequence. This approach, together with the Floyd cycle improvement is the one that Maple has built into its factorisation command '`ifactor(n,pollard).`' The modification which I make for my own students is to allow for variable seed and iteration function, using the Floyd cycle finding method. Once again it is my experience that students are really impressed with how effective the method is.

I finish by giving a Maple procedure [26], the final version of the one I teach to my students, which incorporates the general form of Pollard-Floyd, and give two examples of the sort of output one will see on using it.

```
>PF := proc(n, f, seed) # general Pollard-Floyd
  local a, k; a[1] := seed: a[2] := f(a[1]):
  for k from 2 while igcd(n, a[2*k-2]-a[k-1])=1 do
  a[k] := f(a[k-1]) mod n;
  a[2*k] := f(f(a[2*k-2]) mod n) mod n; od:
  if igcd(a[2*k-2] - a[k-1], n) $<$ n then
  lprint(igcd(a[2*k-2]-a[k-1], n),
        `is a proper factor of`, n);
  else lprint(`Try some other seed or function.`)
  fi; end:

>PF(1037, x-> x^2 + 1, 2);

        17 is a proper factor of 1037

>PF(2^32+1, x -> x^2 + 1, 2); # the 5th Fermat number:

        641 is a proper factor of 4294967297
```

[26] An early Maple worksheet of mine on this topic may be found on the internet [Cos].

References

[BS] Bach, E. and Shallit, J.: **Algorithmic Number Theory. Volume 1**. The MIT Press. (1996)

[B] Bressoud, D. M.: **Factorization and primality testing**. Springer-Verlag. (1989)

[BLSTW] Brillhart, J., Lehmer, D. H., Selfridge, J. L., Tuckerman, B. and Wagstaff, Jr., S. S.: Factorizations of $b^n \pm 1$ ($b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers), AMS (Contemporary Mathematics Series), Vol. 22, 2nd edition, 1988.

[C] Cohen, H.: **A Course in Computational Algebraic Number Theory**. Springer-Verlag. (1993)

[Cos] Cosgrave, J. B.: Several of my Maple worksheets relating to my NT and Cryptography course, including the substantial [27] public lecture of 25th November 1998, Bill Clinton, Bertie Ahern [28], and digital signatures, are accessible from David Joyner's USNA Web site at this address:
`http://web.usna.navy.mil/~wdj/crypto.htm`
At the time of preparing this paper my own web site
`http://www.spd.dcu.ie/johnbcos`
is under construction, and when that is completed I will be putting up a considerable number of my Maple worksheets on that site. Alternatively, please contact me at my College using `John.Cosgrave@spd.ie`, or at home `johnbcos@iol.ie`.

[DH] Diffie, W. and Hellman, M.E.: New Directions in Cryptography. IEEE Transactions on Information Theory, v. IT-22, n. 6, (Nov 1976) 109-112

[K] Kahn, D.: **The Codebreakers (The Comprehensive History of Secret Communication from Ancient Times to the Internet)** (1996) Scribner

[Ko] Koblitz, N.: **A Course in Number Theory and Cryptography**. Springer-Verlag. (1994)

[LT] Lenstra, H. W. and Tijdeman, R. (Editors): **Computational Methods in Number Theory**. Math. Centre Tracts 154 Mathematisch Centrum. Amsterdam. (1982)

[Pol1] Pollard, J. M.: Theorems on Factorization and Primality Testing. Proc. Camb. Phil. Soc. 76 (1974) 521-528

[Pol2] Pollard, J. M.: A Monte Carlo Method for Factorization. BIT. 15 (1975) No. 3. 331-335.

[P1] Pomerance, C.: **Cryptology and Computational Number Theory**. Mathematical Association of America. MAA Notes. 4 (1984)

[P2] Pomerance, C. (Editor): **Cryptology and Computational Number Theory**. American Mathematical Society. Proceedings of Symposia in Applied Mathematics. 42 (1990)

[P3] Pomerance, C.: A Tale of Two Sieves. Notices of the American Mathematical Society. 43 No. 12. (1996) 1473-1485

[Ri] Riesel, H.: **Prime Numbers and Computer Methods for Factorization**. Birkhäuser. (1994)

[27] Over forty pages of hard copy.

[28] The Irish Prime Minister who, in September 1998, participated in an 'historic' digital signing in Ireland of a treaty on e-commerce with the US President, using software developed by the Irish company Baltimore (see their Web site for details of that signing at `http://www.baltimore.ie`)

[RSA] Rivest, R.L., Shamir, A., and Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM. 21(2), (1978) 120-126

[R] Rosen, K. H.: **Elementary Number Theory and Its Applications**. Addison-Wesley. (1988)

email: John.Cosgrave@spd.ie, johnbcos@iol.ie