Implementation of the RSA Public-Key Cryptosystem *

Tham Kine Thong
St Andrew's Junior College

Rainer Schulz
Department of Mathematics
National University of Singapore

§1. Introduction

The history of secret messages, cryptosystems, codes and codecracking is as old as the history of man. In this article, we will deal with the cryptosystem proposed by Rivest, Shamir and Adleman in 1977, see [3]. It has two characteristics which distinguish it from most other encryption methods: The usage of Number Theory and the fact that the process of deciphering is not just opposite to the one of enciphering. As a consequence of the latter feature, the enciphering key is made public, giving the system the attribute "Public-Key". Two applications of this feature will be discussed later, namely a multiuser operation option and an option for authorizing a message by giving a signature.

In section 2, we will provide the background from elementary Number Theory, as far as needed for the following. In section 3, we will describe the RSA cryptosystem. A numerical example will be given in section 4. Section 5 contains a discussion about the safety of the system, section 6 describes the extension of the system to multiuser operation. Finally, in section 7, two computer programs are presented.

The reader will notice that the problems of finding large prime numbers and factorizing large numbers are closely connected with the concept of the RSA cryptosystem. Roughly speaking, the party that wants to transmit a secret message has to find two large prime numbers and use them in the enciphering and deciphering process, whereas the enemy who wants to crack the code of the message has to find the factorization of the product of these two primes. Up to the present day, it is very hard to find large prime numbers. But it is still much harder to factorize large numbers. Hence, the transmitters of the secret message have an advantage

---

* **This article is based on the work done by the first author under the supervision of the second author as part of the 1991 Science Rsearch Programme [4].**

over the enemy. Due to this fact, the RSA cryptosystem has found many applications in business, diplomatic services and, of course, in the military domain.

The reader interested in prime numbers is referred to Ribenboim's beautiful book [1].

§2. Background for Number Theory

For a positive integer m, let $\phi(m)$ denote the number of integers between 1 and m which are relatively prime to m. The function ϕ is called Euler's ϕ-function. Without proof, we recall Euler's Theorem:

Theorem (Euler). *For every integer x relatively prime to m, one has $x^{\phi(m)} \equiv 1 \bmod m$.*

From now on, let p and q be two fixed different odd prime numbers, and let $m = pq$. Note that $\phi(m) = (p-1)(q-1)$. The following result which is an easy consequence of Euler's Theorem is crucial for the RSA cryptosystem.

Theorem. *Let k be an integer with $k \equiv 1 \bmod \phi(m)$. Then, for every integer x, one has $x^k \equiv x \bmod m$.*

Proof. Case 1: p divides x, and q divides x. Then $m = pq$ divides x, and both, x^k and x, are $\equiv 0 \bmod m$.

Case 2: p does not divide x, and q does not divide x. Then m and x are relatively prime, and we have $x^{\phi(m)} \equiv 1 \bmod m$, by Euler's Theorem. From $k \equiv 1 \bmod \phi(m)$ it follows that $k-1$ is a multiple of $\phi(m)$, hence $x^{k-1} \equiv 1 \bmod m$. The assertion follows by multiplication with x, on both sides of the congruence.

Case 3: x is divided by exactly one of the two primes, say q divides x, and p does not. Note that $x^{p-1} \equiv 1 \bmod p$, by Fermat's Theorem (which is a special case of Euler's Theorem). From $k \equiv 1 \bmod \phi(m)$ it follows that $k-1$ is a multiple of $\phi(m)$, say $\phi(m)t = k-1$ with an integer t. Then $(p-1)(q-1)t+1 = k$, hence $x^k \equiv x^{(p-1)(q-1)t+1} \equiv x^{(p-1)(q-1)t}x \equiv x \bmod p$. Combining with $x^k \equiv x \bmod q$ (which is trivial since q divides x), and exploiting the fact that p and q are different, we get $x^k \equiv x \bmod pq$, as desired.

Corollary. Let e and d be integers with $ed \equiv 1 \bmod \phi(m)$. Then, for every integer x, one has $x^{ed} \equiv x \bmod m$.

Remark. In order to find a pair of integers e and d with $ed \equiv 1 \bmod \phi(m)$, one has to pick out any e which is relatively prime to $\phi(m)$. Then one can use Euclid's Algorithm to find integers r and s with $er+\phi(m)s = 1$ and choose d to be the remainder of r modulo $\phi(m)$.

§3. The RSA Cryptosystem

We will use the notations from the preceding section.

Assume a message is given in word form. We can convert it into a sequence of digits using the rule $a = 01$, $b = 02, ..., z = 26$, space $=$ 00. Break this sequence into blocks of length ℓ, say. The blocks must be numbers smaller than m, to ensure that the message is treated "faithfully", in the following.

Enciphering the message: For each block x, form the number x^e and its remainder y modulo m. The numbers y represent the encoded text.

Deciphering: For every number y from the encoded text, form the number y^d and its remainder modulo m.

Then, by the Corollary given in section 2, $y^d \equiv x^{ed} \equiv x \bmod m$. The original text has been recovered.

Remark. Here, we are not assuming that x and m are relatively prime. It is advantageous to make this assumption when one enters a detailed discussion about the safety of the system.

§4. An Example

Let $p = 653$ and $q = 733$. Then $m = 478649$ and $\phi(m) = 477264$. Let $e = 619$. Apply Euclid's Algorithm to find $619*127219+477264*(-165) = 1$. Hence, $d = 127219$.

Let the given message be "PEACE". This message corresponds to the sequence 160501030500. Break it up into the 6-blocks 160501 and 030500. (Note that, in any message, the largest possible block number will be $262626 < m$. Numbers less than 6 digits long may be filled up with zeros.)

Enciphering: $160501^{619} \equiv 413056 \bmod m$, $030500^{619} \equiv 296584 \bmod m$. The encoded text is 413056 296584.

Deciphering: $413056^{127219} \equiv 160501 \bmod m$, $296584^{127219} \equiv 030500 \bmod m$. The original text has been recovered.

Remark. To handle these large powers easily, one can proceed as follows:

(i) Form powers of x with exponents $2, 4, 8, 16, \ldots$, successively, reducing modulo m, after each step.

(ii) Find the binary representation $619 = e_0 + 2e_1 + 4e_2 + 8e_3 + \ldots$ ($e_i = 0$ or 1).

(iii) Using (i) and (ii), form $x^{619} \equiv x^{e_0} x^{2e_1} x^{4e_2} x^{8e_3} \ldots \bmod m$, successively, reducing modulo m, after each step.

§5. The Question of Safety

For reasons to be given later, the numbers m and e are made public. The number d is known only to the receiver of the message. And nobody knows p and q, not even sender or receiver.

Remark 1. An enemy who is able to factorize m can crack the encoded message. This is because knowledge of p and q means knowledge of $(p-1)(q-1) = \phi(m)$. And from e and $\phi(m)$, the enemy can find d easily by applying Euclid's Algorithm, as it was described at the end of section 2. Note that d (when between 1 and $\phi(m)$) is uniquely determined by e, since it is the inverse to e in the group Z_m^* of the prime residues modulo m.

In practice, primes with about 100 digits are regarded to be safe since m would then be 200 digits long, and there is no way to factorize such a large number within a reasonably short time, provided the number is not very special. The numbers $p-1$ and $q-1$ should both contain a large prime factor p' and q', since otherwise fast factorization methods might be successfully applied to m.

Remark 2. Note that knowledge of $\phi(m)$ (and m) is equivalent to knowledge of p and q. This is because $p + q = m - \phi(m) + 1$, and then $p - q = \sqrt{(p+q)^2 - 4m}$, and from $p+q$ and $p-q$ one can recover p and q.

Remark 3. In principle, the enemy can crack the encoded message by taking an encoded message number y and form the powers y^e, y^{2e}, y^{3e} successively, until he obtains a text which makes sense in plain text.

By the Corollary given in section 2, this will happen for sure because some power e^h of e will eventually be congruent 1 modulo $\phi(m)$, namely when h is equal to the order of e in the group Z_m^*. A necessary condition to get a safe code is to make sure that this does not happen so soon, this means one should work with an e having a huge order.

But the latter condition does not yet guarantee a safe code. In the following example, the order of e is greater than 1, but the enciphering is as bad as possible:

Let $p = 43$, $q = 71$, $e = 211$ (then $m = 3053$, $\phi(m) = 2940$, $d = 2731$). Let $\ell = 4$. When forming the powers of any message number x, one will observe that there is actually no enciphering! This follows from $42 = 2 \cdot 3 \cdot 7$, $70 = 2 \cdot 5 \cdot 7$, and consequently the exponent of Z_m^* is $2 \cdot 3 \cdot 5 \cdot 7 = 210$ which implies that the procedure of forming 211th powers does not change x.

Hence, not only the order of e modulo $\phi(m)$ but even the order of e modulo the exponent of Z_m^* must be huge. In particular, the greatest common divisor of $p-1$ and $q-1$ has to be small to ensure the existence of such an e.

Rivest has shown that p and q should be chosen as follows: $p = ap'+1$, $p' = bp''+1$, $q = cq'+1$, $q' = dq''+1$, where p', p'', q', q'' are primes and a, b, c, d are small integers, see [2], for details.

The question of how to find such primes p and q is of special interest, in this context.

§6. Multiuser Operation and Signature

The RSA cryptosystem can be extended to n users U_i $(1 \le i \le n)$ as follows. For every user, find primes p_i and q_i. Form $m_i = p_i \cdot q_i$, choose e_i and find d_i, for every i. Make all of the numbers m_i and e_i public. If U_i wants to send a message to U_j, he enciphers with e_j. Then only U_j and no other user can decipher the message.

It is also possible for U_i to send a "signed" message to U_j. To do so, U_i does not send x^{e_j} but $x^{e_j d_i}$. Then U_j (and only U_j) can decipher by forming $x^{e_j d_i e_i d_j}$, using his secret d_j and the public e_i only. Moreover, U_j can be sure that U_i has been the sender since d_i is only known to U_i.

The multiuser and the signature options may be modified. For example, "headquarters" U_1 may be declared by supplying U_1 with the set D of all deciphering numbrs d_i, and subordinate headquarters may be supplied only with subsets of D.

§7. Implementation of the RSA Cryptosystem

The following two BASIC programs for enciphering and for deciphering have been worked out and tested by the first author. The reader who wants to run them should note that one has to use two three-digit primes p and q whose product is greater than 262626. Also, the input message must be in capital letters. It may be convenient to try with the example from section 4 at first.

There are several restrictions of the programs due to constraints inherent to the computer language used. In particular, in practice one would need a multiprecision arithmetic in order to allow larger primes. The reader is encouraged to write his or her own version of the RSA cryptosystem in a more powerful computer language.

Encipher:

```
1     DEFDBL M,N,P,C,R,Y,Z,V,W
2     DIM N(100),W(100),Z(100),D(100),V(100),R(100),E(100),
      E$(100),C(100),F$(150)
5     DIM P(100),F(100),K$(100)
7     CLS
10    INPUT “FIRST PRIME NUMBER”;A
20    INPUT ‘SECOND PRIME NUMBER”;B
30    M=A*B
40    PRINT “THEIR PRODUCT M =”;M
50    N=(A-1)*(B-1)
60    PRINT “THE NO. OF INTEGERS LESS THAN M AND
      HAS A GCD OF 1 WITH M=N=”;N
510   INPUT “NUMBER FOR C”;C
530   REM *** EUCLID’S ALGORITHM TO FIND GCD ***
547   N(1) = N
548   N(2) = C
560   I = 2
565   P(I) = INT(N(I-1)/N(I))
570   I = I + 1
575   N(I) = N(I-2)-N(I-1)*P(I-1)
578   P(I) = INT(N(I-1)/N(I))
580   IF N(I) = 0 THEN GOTO 610
```

```
610     FOR A=1 TO I
620     PRINT A;")"; N(A); "="; N(A+1); "*"; P(A+1); "+"N(A+2)
625     IF N(A+2)=0 GOTO 640
630     NEXT A
640     IF N(A+1)=1 GOTO 700
650     IF N(A+1) <>1 THEN INPUT "THE GCD OF C AND N
        NOT EQUAL 1, PRESS RETURN TO RETRY, Q TO QUIT";C$
660     IF C$="Q" THEN END
670     IF c$=" " THEN GOTO 5
700     X(1)=1
810     Y(1)=-P(I-2)
820     FOR L=1 TO I
840     X(L+1)=Y(L)
850     Y(L+1)=X(L)-(Y(L))*(P(I-L-2))
855     IF N(A-L) = 0, GOTO 890
860     PRINT N(A+1);"=";"(";N(A-L);") * (";X(L);")
        + (";Y(L);") * (";N(A+1-L);")"
865     IF L=I GOTO 890
870     NEXT L
880     GOTO 840
890     PRINT "POSSIBLE VALUE FOR D =";Y(L+1)
920     IF Y(L+1)>0 THEN U=Y(L+1)
930     IF Y(L+1)<0 THEN U=N+Y(L+1)
940     IF Y(L+1)<0 THEN PRINT "SINCE D SHOULD BE GREATER
        THAN 0, THEREFORE SUITABLE VALUE=";N;Y(L+1);"=";N+Y(L+1
1000    INPUT "WORD TO BE ENCODED";K$
1010    S=LEN(K$)
1020    FOR T=1 TO S
1030    F$(T)=MID$(K$,T,1)
1040    PRINT F$(T)
1050    IF F$(T)="A" THEN F(T)=1
1060    IF F$(T)="B" THEN F(T)=2
1070    IF F$(T)="C" THEN F(T)=3
1080    IF F$(T)="D" THEN F(T)=4
1090    IF F$(T)="E" THEN F(T)=5
1100    IF F$(T)="F" THEN F(T)=6
1110    IF F$(T)="G" THEN F(T)=7
1120    IF F$(T)="H" THEN F(T)=8
1130    IF F$(T)="I" THEN F(T)=9
```

```
1140    IF F$(T)="J" THEN F(T)=10
1150    IF F$(T)="K" THEN F(T)=11
1160    IF F$(T)="L" THEN F(T)=12
1170    IF F$(T)="M" THEN F(T)=13
1180    IF F$(T)="N" THEN F(T)=14
1190    IF F$(T)="O" THEN F(T)=15
1200    IF F$(T)="P" THEN F(T)=16
1210    IF F$(T)="Q" THEN F(T)=17
1220    IF F$(T)="R" THEN F(T)=18
1230    IF F$(T)="S" THEN F(T)=19
1240    IF F$(T)="T" THEN F(T)=20
1250    IF F$(T)="U" THEN F(T)=21
1260    IF F$(T)="V" THEN F(T)=22
1270    IF F$(T)="W" THEN F(T)=23
1280    IF F$(T)="X" THEN F(T)=24
1290    IF F$(T)="Y" THEN F(T)=25
1300    IF F$(T)="Z" THEN F(T)=26
1310    IF F$(T)=" " THEN F(T)=0
1320    PRINT F(T)
1330    NEXT T
1340    FOR T=1 TO N STEP 3
1350    W=(F(T)*10000)+(F(T+1)*100)+F(T+2)
1355    IF W=0, THEN END
1360    PRINT W
5010    REM *** TO EXPRESS C IN BINARY ***
5015    C(30)=C
5018    V(30)=INT(C(30)/2^ 30)
5019    F=V(30)*2^ 30
5020    FOR I=29 TO 0 STEP -1
5030    C(I)=C(I+1)-F
5040    V(I)=INT(C(I)/2^ I)
5050    F=V(I)*2^ I
5060    NEXT I
5070    REM *** TO EVALUATE REMAINDER OF A^ (2^ N)
        MODULO M ***
5080    FOR I=1 TO 30
5090    W(0)=W
5100    G=INT(W(I-1)*W(I-1)/M)
5110    W(I)=W(I-1)*W(I-1)-M*G
```

```
5120    NEXT I
5130    REM *** TO EVALUATE W^ [C/2] MODULO M ***
5140    R=1
5150    FOR I=0 TO 30
5160    Z(I)=W(I)*V(I)
5170    Q=0
5180    FOR J=0 TO 30
5190    Q=Q+1
5200    IF Z(I)<>0 THEN IF Q=1 THEN D(J)=Z(I)
        ELSE GOTO 5240 ELSE GOTO 5240
5210    H=INT(R*D(J)/M)
5220    R=R*D(J)-H*M
5230    NEXT J
5240    NEXT I
5250    PRINT “THE ENCODED MESSAGE=”;R
5260    NEXT T
5270    GOTO 1350
        Decipher:
1       DEFDBL M,N,P,C,R,Y,Z,V,W
3       DIM W(100),Z(100),D(100),V(100)
8       CLS
10      INPUT “VALUE FOR M=”;M
20      INPUT “VALUE FOR D=”;D
30      INPUT “MESSAGE TO BE DECODED (INPUT 0 TO END)=”;W
35      IF W=0 THEN END
40      REM *** TO EXPRESS D IN BINARY ***
42      D(30)=D
48      V(30)=INT(D(30)/2^ 30)
49      Z=V(30)*2^ 30
50      FOR I=29 TO 0 STEP -1
60      D(I)=D(I+1)-Z
70      V(I)=INT(D(I)/2^ I)
80      Z=V(I)*2^ I
90      NEXT I
100     REM *** TO EVALUTE REMAINDER OF A^ (2^ N)
        MODULO M ***
110     FOR I=1 TO 30
120     W(0)=W
130     G=INT(W(I-1)*W(I-1)/M)
```

```
140   W(I)=W(I-1)*W(I-1)-M*G
150   NEXT I
160   REM *** TO EVALUATE W^ [C/2] MODULO M ***
170   R=1
180   FOR I=0 TO 30
190   Z(I)=W(I)*V(I)
200   Q=0
210   FOR J=0 TO 30
220   Q=Q+1
230   IF Z(I)<>0 THEN IF Q=1 THEN D(J)=Z(I)
      ELSE GOTO 270 ELSE GOTO 270
240   H=INT(R*D(J)/M)
250   R=R*D(J)-H*M
260   NEXT J
270   NEXT I
280   PRINT "THE DECODED MESSAGE =";R
290   C=INT(R/10000)
300   F=INT((R-C*10000)/100)
310   L=INT(R-C*10000-F*100)
320   PRINT C,F,L
330   IF C=1 THEN C$="A"
340   IF C=2 THEN C$="B"
350   IF C=3 THEN C$="C"
360   IF C=4 THEN C$="'D"
370   IF C=5 THEN C$="E"
380   IF C=6 THEN C$="F"
390   IF C=7 THEN C$="G"
400   IF C=8 THEN C$="H"
410   IF C=9 THEN C$="I"
420   IF C=10 THEN C$="J"
430   IF C=11 THEN C$="K"
440   IF C=12 THEN C$="L"
450   IF C=13 THEN C$="M"
460   IF C=14 THEN C$="N"
470   IF C=15 THEN C$="O"
480   IF C=16 THEN C$="P"
490   IF C=17 THEN C$="Q"
500   IF C=18 THEN C$="R"
510   IF C=19 THEN C$="S"
```

```
520    IF C=20 THEN C$="T"
530    IF C=21 THEN C$="U"
540    IF C=22 THEN C$="V"
550    IF C=23 THEN C$="W"
560    IF C=24 THEN C$="X"
570    IF C=25 THEN C$="Y"
580    IF C=26 THEN C$="Z"
590    IF C=0 THEN C$=" "
630    IF F=1 THEN F$="A"
640    IF F=2 THEN F$="B"
650    IF F=3 THEN F$="C"
660    IF F=4 THEN F$="D"
670    IF F=5 THEN F$="E"
680    IF F=6 THEN F$="F"
690    IF F=7 THEN F$="G"
700    IF F=8 THEN F$="H"
710    IF F=9 THEN F$="I"
720    IF F=10 THEN F$="J"
730    IF F=11 THEN F$="K"
740    IF F=12 THEN F$="L"
750    IF F=13 THEN F$="M"
760    IF F=14 THEN F$="N"
770    IF F=15 THEN F$="O"
780    IF F=16 THEN F$="P"
790    IF F=17 THEN F$="Q"
800    IF F=18 THEN F$="R"
810    IF F=19 THEN F$="S"
820    IF F=20 THEN F$="T"
830    IF F=21 THEN F$="U"
840    IF F=22 THEN F$="V"
850    IF F=23 THEN F$="W"
860    IF F=24 THEN F$="X"
870    IF F=25 THEN F$="Y"
880    IF F=26 THEN F$="Z"
890    IF F=0 THEN F$=" "
930    IF L=1 THEN L$="A"
940    IF L=2 THEN L$="B"
950    IF L=3 THEN L$="C"
960    IF L=4 THEN L$="D"
```

```
970   IF L=5 THEN L$=“E”
980   IF L=6 THEN L$=“F”
990   IF L=7 THEN L$=“G”
1000  IF L=8 THEN L$=“H”
1010  IF L=9 THEN L$=“I”
1020  IF L=10 THEN L$=“J”
1030  IF L=11 THEN L$=“K”
1040  IF L=12 THEN L$=“L”
1050  IF L=13 THEN L$=“M”
1060  IF L=14 THEN L$=“N”
1070  IF L=15 THEN L$=“O”
1080  IF L=16 THEN L$=“P”
1090  IF L=17 THEN L$=“Q”
1100  IF L=18 THEN L$=“R”
1110  IF L=19 THEN L$=“S”
1120  IF L=20 THEN L$=“T”
1130  IF L=21 THEN L$=“U”
1140  IF L=22 THEN L$=“V”
1150  IF L=23 THEN L$=“W”
1160  IF L=24 THEN L$=“X”
1170  IF L=25 THEN L$=“Y”
1180  IF L=26 THEN L$=“Z”
1190  IF L=0 THEN L$=“ ”
1200  PRINT C$,F$,L$
1210  GOTO 30
```

References

[1] P. Ribenboim, *The Book of Prime Number Records,* Springer-Verlag, New York 1988.

[2] R. L. Rivest, Remarks on a proposed cryptoanalytic attack on the M.I.T. public-key cryptosystem, *Cryptologia 2,* 1978.

[3] R. L. Rivest, A. Shamir and L. M. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Comm. ACM 21,* 1978, 120-126.

[4] Tham Kine Thong and R. Schulz, The RSA public key cryptosystem, *Proceedings of Science Research Congress 1991,* Singapore 1991.