Subject: How to crack Netrek RSA binary verification system
Date: September 19, 2004
From: J Corrales

The RSA Algorithm
Select two primes p and q
Calculate n = p q
Calculate f(n) = (p-1)(q-1)
Select e such that 1 < e < f(n) and gcd(f(n),e) = 1
Calculate d = e^-1 mod f(n)
Public key KU = {e,n}
Private key KR = {d,n}

Example

Select two primes p=7 and q=17
Calculate n = p q = 119
Calculate f(n) = (p-1)(q-1) = 96
Select e such that 1 < e < f(n) and gcd(f(n),e) = 1, e.g., e = 5
Calculate d = e^-1 mod f(n), e.g., d = 77
Public key KU = {e,n} = {5,119}
Private key KR = {d,n} = {77,119}

Cracking RSA

Factor n, which is public, yielding p and q
Calculate f(n) = (p-1)(q-1)
Calculate d = e^-1 mod f(n) (e is public)
Private key KR = {d,n}
Note: you can check your calculation of d because d e = 1 mod f(n)

Cracking RSA (Example)

Factor 119, which is public, yielding 7 and 17
Calculate f(119) = (7-1)(17-1) = 96
Calculate 5^-1 = 77 mod 96
Private key KR = {77,119}
Plaintext M = 19
Ciphertext C = Me mod n = 195 mod 119 = 66
Plaintext M = Cd mod n = 6677 mod 119 = 19

Netrek RSA Specifics:

Go to http://65.193.17.240:3530/ and select a key. Try:

```
key.brmh2.sun4:ct=BRM-Hadley:cr=hadley@uci.edu:\
   :cd=September 1994:ar=Sun4 / SunOS 4.1.2:cl=inl,standard2:\
   :cm=At cad.ics.uci.edu-/pub/netrek:\
   :gk=636a5977da5ce59948885d1003e983deabf405ed0cb952a5b42930523909f901:\
   :pk=0dfc24f1d19398586dafa37e985163290eb01f139c03f90affc5d458a7f6b800:
```

Convert gk and pk to integers. Note gk and pk are little endian, so
they must be reversed and converted. I used java:

```
//be2i.java : converts gk and pk to plain integers
//
import java.math.BigInteger;

public class be2i {
    static BigInteger GLOBAL_KEY = new
BigInteger(swapEndian("636a5977da5ce59948885d1003e983deabf405ed0cb952a5b42930523909f901"),
16);
    static BigInteger PUBLIC_KEY = new
BigInteger(swapEndian("0dfc24f1d19398586dafa37e985163290eb01f139c03f90affc5d458a7f6b800"),
16);

    static String swapEndian(String s)
   {
       char c[] = s.toCharArray();
       StringBuffer buffer = new StringBuffer(c.length);
       for(int i = c.length; (i -= 2) >= 0;)
       {
           buffer.append(c[i]);
           buffer.append(c[i + 1]);
       }

       return buffer.toString();
   }

    public static void main(String args[]) {
        System.out.println(GLOBAL_KEY.toString(10));
        System.out.println(PUBLIC_KEY.toString(10));
        }
}
```

```
E:\be2i>javac -d . be2i.java

E:\be2i>java be2i
89232142880258621900150271977837845127167771696140333372458652965941080329 9
32680220118663818558764939273917102572684654235632805626767521353197400782 1
```

Note:

```
gk=89232142880258621900150271977837845127167771696140333372458652965941080329 9
pk=32680220118663818558764939273917102572684654235632805626767521353197400782 1
```

Factor gk. Download PARI/GP http://pari.math.u-bordeaux.fr/ Install
it, then run it:

```
Reading GPRC: /cygdrive/c/Program Files/PARI/.gprc ...Done.

          GP/PARI CALCULATOR Version 2.2.8 (development CHANGES-1.887)
               i686 running cygwin (ix86 kernel) 32-bit version
              compiled: Jan 13 2004, gcc-3.3.1 (cygming special)
               (readline v4.3 enabled, extended help available)

                     Copyright (C) 2003 The PARI Group

PARI/GP is free software, covered by the GNU General Public License,
and
comes WITHOUT ANY WARRANTY WHATSOEVER.
```

```
Type ? for help, \q to quit.
Type ?12 for how to get moral (and possibly technical) support.

   realprecision = 28 significant digits
   seriesprecision = 16 significant terms
   format = g0.28

parisize = 4000000, primelimit = 500000
(19:29) gp > factorint(89232142880258621900150271977837845127167771696140333372458652965941080329 9)

  ***   Warning: MPQS: the factorization of this number will take
several hours.

%1 =
[6158427920916659824908651606478252183 1]

[1448943529519733969370375319274588472 53 1]

(21:29) gp >
```

Calculate d = e^-1 mod f(n) (e is public)

Download res-rsa-2.9.2
ftp://ftp.netrek.org/pub/netrek/rsa/res-rsa-2.9.2.tar.gz

Modify mkkey.c as follows:

```
    /*    mpz_set_ui(x, 0); */
    /* mpz_set_ui(y, 0); */
    /* mpz_set_ui(global, 0); */

    mpz_set_str(x,"6158427920916659824908651606478252183", 0);
    mpz_set_str(y, "1448943529519733969370375319274588472 53", 0);
    mpz_set_str
      (global, "89232142880258621900150271977837845127167771696140333372458652965941080329 9",
0);
    mpz_set_str
      (private, "32680220118663818558764939273917102572684654235632805626767521353197400782 1",
0);
    /* mpz_set_ui(public, 0); */
    mpz_set_ui(public, 0);
    mpz_set_ui(xyminus1, 0);

    /* here we find x and y, two large primes */

    rand_raw(temp, HALF);
    temp[0] |= 1; /* force odd */
    /* raw_to_num(x, temp); */

    while (!is_prime(x))
      mpz_add(x, x, two);

    check_positive(x);
    assert(is_prime(x));

    rand_raw(temp, HALF);
    temp[0] |= 1; /* force odd */
    /* raw_to_num(y, temp); */

    while (!is_prime(y))
      mpz_add(y, y, two);

    check_positive(y);
    assert(is_prime(y));

    /* the private key is a large prime (it should be the larger than
     * x & y) */

    rand_raw(temp, HALF + 1);
    temp[0] |= 1; /* force odd */
    //raw_to_num(private, temp);

    //    while (!is_prime(private))
    //      mpz_add(private, private, two);

    check_positive(private);
    //assert(is_prime(private));
```

Note rest of mkkeys.c is unchanged.

```
cherry:/home/bd/netrek/mkkey/res-rsa-2.9.2$ diff mkkey.c mkkey-mine.c
1103,1106c1119,1129
<     mpz_set_ui(x, 0);
<     mpz_set_ui(y, 0);
<     mpz_set_ui(global, 0);
<     mpz_set_ui(private, 0);
---
>     /*    mpz_set_ui(x, 0); */
>     /* mpz_set_ui(y, 0); */
>     /* mpz_set_ui(global, 0); */
>
>     mpz_set_str(x,"6158427920916659824908651606478252183", 0);
>     mpz_set_str(y, "1448943529519733969370375319274588472 53", 0);
>     mpz_set_str
>       (global, "89232142880258621900150271977837845127167771696140333372458652965941080329 9", 0);
>     mpz_set_str
>       (private, "32680220118663818558764939273917102572684654235632805626767521353197400782 1", 0);
>     /* mpz_set_ui(public, 0); */
1114c1137
<     raw_to_num(x, temp);
---
>     /* raw_to_num(x, temp); */
1124c1147
<     raw_to_num(y, temp);
---
>     /* raw_to_num(y, temp); */
1137c1160
<     raw_to_num(private, temp);
---
>     //raw_to_num(private, temp);
1139,1140c1162,1163
<     while (!is_prime(private))
<       mpz_add(private, private, two);
---
>     //    while (!is_prime(private))
>     //      mpz_add(private, private, two);
```

```
1143c1166
<     assert(is_prime(private));
---
>     //assert(is_prime(private));

cherry:/home/bd/netrek/mkkey/res-rsa-2.9.2$ make
gcc -O2  -c mkkey.c
gcc -O2  -o mkkey mkkey.o -lgmp
cherry:/home/bd/netrek/mkkey/res-rsa-2.9.2$ ./mkkey foo foo foo foo foo
foo
mkkey version [RES-RSA 2.9.2: Mar. 13, 2000][GMP]
Source basename: "rsa_box"
Number of shells: 3
Number of steps between swaps: 2
Number of files: 5
Ratio of computation in files to main file: 0.8
Making new key, hold on....
Testing key ..................................................... key
seems OK
Writing...
240 bits left, 5 files left, 37 bits in rsa_box_0.c
202 bits left, 4 files left, 7 bits in main file
194 bits left, 4 files left, 42 bits in rsa_box_1.c
151 bits left, 3 files left, 11 bits in main file
139 bits left, 3 files left, 29 bits in rsa_box_2.c
109 bits left, 2 files left, 10 bits in main file
98 bits left, 2 files left, 32 bits in rsa_box_3.c
65 bits left, 1 files left, 13 bits in main file
51 bits left, 1 files left, 39 bits in rsa_box_4.c

cherry:/home/bd/netrek/mkkey/res-rsa-2.9.2$ cat foo.secret
foo:ct=foo:cr=foo:\
   :cd=September 2004:ar=foo:\
   :cm=foo:\
   :gk=636a5977da5ce59948885d1003e983deabf405ed0cb952a5b42930523909f901:\
   :pk=057505d6c5b0b5ccd0f7819a7670da469e00000000000000000000000000000:\
   :sk=0dfc24f1d19398586dafa37e985163290eb01f139c03f90affc5d458a7f6b800:

Note pk and sk are reversed. Edit foo.secret to correct the order to:

foo:ct=foo:cr=foo:\
   :cd=September 2004:ar=foo:\
   :cm=foo:\
   :gk=636a5977da5ce59948885d1003e983deabf405ed0cb952a5b42930523909f901:\
   :pk=0dfc24f1d19398586dafa37e985163290eb01f139c03f90affc5d458a7f6b800:\
   :sk=057505d6c5b0b5ccd0f7819a7670da469e00000000000000000000000000000:

Rerun mkkey with the edited secret key file:

cherry:/home/bd/netrek/mkkey/res-rsa-2.9.2$ ./mkkey -k foo.secret
mkkey version [RES-RSA 2.9.2: Mar. 13, 2000][GMP]
Source basename: "rsa_box"
Number of shells: 3
Number of steps between swaps: 2
Number of files: 5
Ratio of computation in files to main file: 0.8
Reading key from keycap file "foo.secret"...
Testing key ..................................................... key
seems OK
Writing...
128 bits left, 5 files left, 19 bits in rsa_box_0.c
108 bits left, 4 files left, 6 bits in main file
101 bits left, 4 files left, 20 bits in rsa_box_1.c
80 bits left, 3 files left, 5 bits in main file
74 bits left, 3 files left, 15 bits in rsa_box_2.c
58 bits left, 2 files left, 6 bits in main file
51 bits left, 2 files left, 21 bits in rsa_box_3.c
29 bits left, 1 files left, 5 bits in main file
23 bits left, 1 files left, 14 bits in rsa_box_4.c

cherry:/home/bd/netrek/mkkey/res-rsa-2.9.2$ ls -aFl rsa_box_?.c
-rw-r--r--    1 bd     bd         3856 Sep 19 20:02 rsa_box_0.c
-rw-r--r--    1 bd     bd         4093 Sep 19 20:02 rsa_box_1.c
-rw-r--r--    1 bd     bd         3222 Sep 19 20:02 rsa_box_2.c
-rw-r--r--    1 bd     bd         4368 Sep 19 20:02 rsa_box_3.c
-rw-r--r--    1 bd     bd         3130 Sep 19 20:02 rsa_box_4.c
cherry:/home/bd/netrek/mkkey/res-rsa-2.9.2$
```

Copy rsa_box_?.c files into your own custom client source directory,
compile, and login to any public netrek server, including INL server
of your choice.