

```
1 #include <lnv3/lnv3.h>
2 #include <nttl/randomPrime.h>
3 #include <nttl/gcd.h>
4 #include <nttl/inverse.h>
5 #include <time.h>
6
7 #define KEY_SIZE 308
8
9 struct PrivateRsaKey
10 {
11     ln N;
12     ln e;
13     ln d;
14 };
15
16 struct Party
17 {
18     PrivateRsaKey key;
19 };
20
21 void CompleteRsa( PrivateRsaKey& party, ln phiN );
22
23 #define PerformRsa( y, N, ed ) y.FastExp( ed, N )
24
25 void main()
26 {
27     Party alice, bob;
28
29     srand( time( 0 ) );
30
31     // Generate N
32     // In real life you would use Diffie-Helman to have
33     // Alice and Bob determine P and Q without actually
34     // sending them or N over the network
35     ln p, q;
36     ln N;
37     ln phiN;
38     RandomPrime( &p, KEY_SIZE / 2 );
39     RandomPrime( &q, KEY_SIZE / 2 );
40     N = p * q;
41     phiN = ( p - 1 ) * ( q - 1 );
42     alice.key.N = N;
43     bob.key.N = N;
44
45     // Alice and Bob generate their own e/d
46     CompleteRsa( alice.key, phiN );
47     CompleteRsa( bob.key, phiN );
48
49     // Alice generates heads/tails message
50     ln heads = ln().Random( KEY_SIZE ) % alice.key.N;
51     ln tails = ln().Random( KEY_SIZE ) % alice.key.N;
52
53     // Alice encrypts and sends heads/tails messages in a random order
54     ln eheads = PerformRsa( heads, alice.key.N, alice.key.e );
55     ln etails = PerformRsa( tails, alice.key.N, alice.key.e );
56     ln msgs[ 2 ];
57     if( rand() % 2 == 0 )
58     {
59         msgs[ 0 ] = eheads;
60         msgs[ 1 ] = etails;
61     }
62     else
63     {
64         msgs[ 1 ] = eheads;
65         msgs[ 0 ] = etails;
66     }
67 }
```

```
68 // Bob picks one of the messages at random and sends encryption to Alice
69 ln bobChoice = msgs[ rand() % 2 ];
70 ln ebobChoice = PerformRsa( bobChoice, bob.key.N, bob.key.e );
71
72 // Alice gets encrypted, and does her little shuffle, and sends back
73 ln midway = PerformRsa( ebobChoice, alice.key.N, alice.key.d );
74
75 // Bob decrypts and sends back to Alice
76 ln bobFinal = PerformRsa( midway, bob.key.N, bob.key.d );
77
78 // Alice verifies that it matches on of her strings
79 if( bobFinal == heads )
80     cout << "Bob picked heads" << endl;
81 else if( bobFinal == tails )
82     cout << "Bob picked tails" << endl;
83 else
84     cout << "Someone cheated!" << endl;
85
86 // Alice and Bob share private keys and verify
87 // ...
88
89 return;
90 }
91
92 void CompleteRsa( PrivateRsaKey& party, ln phiN )
93 {
94     while( true )
95     {
96         party.e = ln().Random( KEY_SIZE ) % phiN;
97         if( GCD( party.e, phiN ) == 1 )
98             break;
99     }
100     party.d = Inverse( party.e, phiN );
101 }
102
```