

# Split Generation of RSA Parameters with Multiple Participants

Clifford Cocks

## Abstract

We show how it is possible for any number of parties to co-operate in generating the parameters for an RSA encryption system in such a way that all must co-operate in order to decrypt enciphered data. This paper extends previous work which solved this problem for two users. In addition we show how the new protocol can be used to allow arbitrarily large encryption exponents.

## 1. INTRODUCTION

In earlier work [1], we showed that it is possible for two parties to co-operate in generating the parameters for an RSA encryption system in such a way that neither individually has the ability to decrypt enciphered data. In order to decrypt data the two parties had to follow a specific co-operative procedure. In this paper, we extend the result to allow an arbitrary number of co-operating parties. Also, we show how the new ideas can be used to allow arbitrarily large encryption exponents.

## 2. BASIC ALGORITHM

Central to the new method is the following capability. Suppose that there are  $n$  co-operating parties, where each party  $i$  is in possession of two large positive integers  $p_i$  and  $q_i$ . Then there is a procedure such that each party  $i$  calculates a value  $N_i$ , with the following properties:

a.  $\sum_i N_i = 2(p_1 + p_2 + \dots + p_n)(q_1 + q_2 + \dots + q_n) = 2N$ , say.

b. If party  $i$  reveals  $N_i$ , and all other participants collaborate, they are unable to determine  $p_i$  and  $q_i$  with a feasible amount of work, as long as factorisation of the number  $N$  is infeasible, and as long as  $p_i$  and  $q_i$  are large enough.

c. If party  $i$  does not reveal  $N_i$ , then even if they collaborate, the other parties cannot determine  $N$  with feasible work, assuming that at least one of  $p_i$  and  $q_i$  are large enough that their values cannot be exhaustively guessed.

## 2.1 Description of the algorithm

The algorithm works as follows:

a. Initially each participant  $i$  produces their own RSA modulus  $M_i$  and public exponent  $e_i$ . The modulus should be at least as big as 2 times  $N_L$ , the largest possible value of  $N = (p_1 + p_2 + \dots + p_n)(q_1 + q_2 + \dots + q_n)$  that may be produced.

b. Each participant also produces a set of additives  $a_{ij}$ , for  $j = 1, 2, \dots, n$ , where the  $a_{ij}$  are randomly chosen in the range  $0 < a_{ij} < N_L$ , unless  $i = j$ . These numbers must satisfy the equation  $\sum_j a_{ij} = 0$ . Thus  $a_{ii}$  will be negative.

c. Next each participant  $i$  broadcasts  $M_i$ ,  $e_i$ ,  $p_i^{e_i} \bmod M_i$  and  $q_i^{e_i} \bmod M_i$

d. Then each participant  $i$  computes for sending to  $j$  the three numbers:  $(p_i q_j)^{e_j} \bmod M_j$ ,  $(q_i p_j)^{e_j} \bmod M_j$  and  $a_{ij}^{e_j} \bmod M_j$

e. However, before sending these numbers to  $j$ , each is split into  $K$  parts using the same technique as in the two party case. In detail, suppose that  $X$  is one of the numbers to be sent. Then participant  $i$  generates  $K$  numbers  $x_k$ , randomly chosen modulo  $M_j$  and such that  $\sum_k x_k = 1 \bmod M_j$ . Then the  $K$  parts that  $i$  sends to  $j$  are  $X x_k^{e_j} \bmod M_j$ . The  $3K$  values are sent in a new order such that  $j$  does not know which parts corresponds to each of the three numbers that  $i$  is sending. The security of the scheme depends on making  $K$  large enough, and the same considerations apply as in the two party case.

f. By decrypting and summing the RSA encrypted data sent to him, each participant  $i$  will have received enough information to enable him to calculate the following for every  $j \neq i$ :

$$p_i q_j + p_j q_i + a_{ji} \bmod M_i$$

But because  $M_i$  was chosen to be sufficiently large, this must equal:

$$p_i q_j + p_j q_i + a_{ji}$$

Hence by summing these values, and adding in the term  $2p_i q_i + a_{ii}$  that he knows himself, participant  $i$  can calculate

$$N_i = \sum_j p_i q_j + \sum_j p_j q_i + \sum_j a_{ji}$$

g. By sharing  $N_i$ , all parties will be able to determine the sum of these quantities:

$$\sum_i N_i = 2(p_1 + p_2 + \dots + p_n)(q_1 + q_2 + \dots + q_n).$$

Alternatively, by sharing only  $N_i \bmod M$ , for some commonly agreed odd modulus  $M$ , the participants will be able to determine  $N \bmod M$ .

### 3. GENERATION OF RSA PARAMETERS

In this section we show how to use the basic algorithm to enable the participants to set up the parameters for an RSA system, in such a way that only by co-operating can they decrypt data. The three steps in this process are: generating a common modulus, testing that the modulus is in fact a product of two primes (to an acceptable degree of certainty), and generating individual shares of the decrypt exponent.

#### 3.1 Generation of Modulus

This uses the basic algorithm described above. Each participant  $i$  generates two positive integers  $p_i$  and  $q_i$  in an agreed range, satisfying any other specific properties as discussed below. Then the participants use the basic algorithm to generate a candidate modulus:

$$N = (p_1 + p_2 + \dots + p_n)(q_1 + q_2 + \dots + q_n).$$

#### 3.2 Primality Testing

There is no particular reason to expect that  $P = (p_1 + p_2 + \dots + p_n)$  and  $Q = (q_1 + q_2 + \dots + q_n)$  are both prime, and so many candidate  $N$  will need to be tested until one is found that is satisfactory. Initial testing is done by selecting random values  $x$ , and having each participant calculate  $x^{p_i+q_i} \bmod N$ . Then if  $N$  is the product of two primes, the product of all these numbers will equal  $x^{N+1} \bmod N$ .

There is still the requirement to eliminate Carmichael numbers which the above test will pass. The Boneh and Franklin test is described in [1] for the two person case and can be readily extended to the multiple participant situation. To use this test it is necessary that  $P$  and  $Q$  are congruent to 3 modulo 4, and thus the participants will have to decide in advance on the values of  $p_i$  and  $q_i$  modulo 4, for every  $i$ .

The process of generating a suitable  $N$  is made more efficient if the numbers  $p_i$  and  $q_i$  are controlled to ensure that neither  $P$  nor  $Q$  is a multiple of the first few primes. This can be done in much the same way as proposed for the two person case, either by agreeing on the value of  $p_i$  and  $q_i$  modulo  $p$  for primes  $p \leq n$ , or by making  $p_i$  and  $q_i$  less than  $\lceil n/p \rceil$  for small primes  $p$  larger than  $n$ .

### 3.3 Generation of Exponent

Firstly, the public exponent  $e$  is agreed. All parties will need to share  $p_i + q_i \bmod e$ , so  $e$  should not be too large. Also,  $P - 1$  and  $Q - 1$  have to be coprime to  $e$  and if they are not then  $N$  has to be rejected or a different value of  $e$  must be selected.

If the decrypt exponent is  $d$ , then  $d$  will satisfy:

$$ed = 1 \bmod (N + 1 - P - Q)$$

Thus:  $ed \equiv 1 + k(N + 1 - (p_1 + p_2 + \dots + p_n) - (q_1 + q_2 + \dots + q_n))$

Here  $k$  is chosen to make the right hand side a multiple of  $e$ , and so  $k$  can be determined by all parties from the shared values of  $p_i$  and  $q_i$  modulo  $e$ .

The individual decrypt exponent, known only by participant  $i$ ,  $d_i$  say, is then determined by:

$$d_i = \lfloor \frac{(1/n) + (k/n)(N + 1) - kp_i - kq_i}{e} \rfloor$$

Hence the decrypt exponent  $d$  will satisfy  $d = (d_1 + d_2 + \dots + d_n) + c$ , where  $c$  will lie between 0 and  $n - 1$ . One trial decryption will be sufficient to identify  $c$ , and all participants may know  $c$ .

Thus to decrypt a message  $X$ , participant  $i$  will calculate  $X^{d_i} \bmod N$ , and the decrypt will be the product of these  $n$  terms together with  $X^c \bmod N$ .

## 4. FURTHER APPLICATION OF THE BASIC ALGORITHM

The basic algorithm can be used to enable arbitrary encrypt exponents to be employed. To do this, I follow the method of Boneh and Franklin in [2], substituting the basic algorithm for their BGW protocol. The method is described below, where  $e$  is the encrypt exponent. In what follows, I assume that  $e$  is large (of the same order as the the modulus  $N$ ) and in this case the moduli  $M_i$  used in the basic algorithm will need to be at least  $3N^2$  in size. The maximum value of the random  $a_{ij}$  terms used in the protocol should be  $N^2$ .

### 4.1 Step 1

We define  $\phi_1 = (N + 1) - p_1 - q_1 \bmod e$  and  $\phi_i = -p_i - q_i \bmod e$ , for  $i > 1$ , which is computable by participant  $i$ .

Each participant  $i$  generates an arbitrary value mod  $e$ ,  $r_i$ , say. Then they execute the basic algorithm, with participant  $i$  using as input values  $r_i$  and  $\phi_i$ . They share their recoveries modulo  $e$ , so that they can all determine:

$$\Psi = \left(\sum_i r_i\right)\left(\sum_i \phi_i\right) \bmod e$$

If  $\Psi = 0$ , they restart with new values of  $r_i$ . If this fails repeatedly then they will need to either choose a different modulus  $N$  or a different exponent  $e$ .

### 4.2 Step 2

Each participant  $i$  now calculates  $\zeta_i = -r_i/\Psi \bmod e$ . Thus

$$\sum_i \zeta_i = -\left(\sum_i r_i\right)\Psi^{-1} = -\left(\sum_i \phi_i\right)^{-1} \bmod e$$

### 4.3 Step 3

Finally the participants use the basic algorithm with inputs  $\zeta_i, \phi_i$  respectively. If the output recovered by participant  $i$  is  $\eta_i$ , then

$$\sum_i \eta_i = \left(\sum_i \zeta_i\right)\left(\sum_i \phi_i\right) \equiv -1 \bmod e$$

Thus the decrypt exponent  $d$  equals  $\frac{1+\sum_i \eta_i}{e}$ . The individual decrypt exponent,  $d_i$ , known only by participant  $i$ , is given by:

$$d_i = \left\lfloor \frac{\eta_i}{e} \right\rfloor$$

Hence the decrypt exponent  $d$  will satisfy  $d = (d_1 + d_2 + \dots + d_n) + c$ , where  $c$  will lie between 0 and  $n - 1$ . One trial decryption will be sufficient to identify  $c$ . Thus as in the small exponent case, to decrypt a message  $X$ , participant  $i$  will calculate  $X^{d_i} \bmod N$ , and the decrypt will be the product of these  $n$  terms together with  $X^c \bmod N$ .

## 4.4 Smaller encryption exponent

It is possible to use the above approach even if  $e$  is much smaller than the generated modulus  $N$ . However, it is necessary to make the following change in order to retain the security of the method. Before using  $\phi_i$ ,  $r_i$  and  $\zeta_i$  in the basic algorithm the participants should add to them random multiples of  $e$ , subject to the values being less than a limit  $L$ . The moduli  $M_i$  used in the basic algorithm will need to be at least  $3L^2$  in size, and the random values  $a_{ij}$  used in the protocol should not exceed  $L^2$ . A value of  $L=N$  is recommended.

## 5. SECURITY ISSUES

D. Coppersmith [3] points out that each participant's public RSA exponent,  $e_i$  needs to be large for the above protocol to be secure. For, suppose that the other participants collaborate. Then they know the following three values:

$$\begin{aligned} N &= (p_i + a)(q_i + b), \text{ where } a \text{ and } b \text{ are known by the collaborating parties.} \\ p_i^{e_i} &\text{ mod } M_i, \text{ and} \\ q_i^{e_i} &\text{ mod } M_i. \end{aligned}$$

Then, by substituting  $q_i = \frac{N}{(p_i+a)} - b$  into the third expression we get two equations of degree  $e_i$  in  $p_i$ , which can be solved by taking the GCD of the two polynomials modulo  $M_i$ . By [4], the work to do this is of order  $e_i(\log^2(e_i) + \log M_i)$  modular operations. This attack also applies to the two party case [1], except that there is no need for the parties to collaborate to execute the attack.

### 5.1 Cheating

Another concern is the possibility that some of the participants will not execute the protocol honestly, but instead will cheat in such a way as to obtain the secret parameters. This has been analysed in [5] for the two party case, and the ideas carry across to the multi party situation, particularly if we allow all but one of the parties to collaborate in an attempt to cheat. In [5] the authors propose a revised protocol to eliminate cheating.

Cheating during the generation of  $N$  whereby one party substitutes an alternative value for a parameter after receiving input from the others, can be prevented by ensuring that at each step every party commits in advance a hash of the data they are to send. Then the data is sent only after all the commitment information has been received, thus preventing it from being modified.

Cheating during the primality test, for example an attempt to falsely convince one party that a number is a product of two primes can be prevented with the ideas in [5]. All parties agree on a large prime  $R$ , for which there is a large prime

factor  $S$  of  $R-1$ . They choose an element  $\alpha$  of order  $S$  in the multiplicative group modulo  $R$ . Before executing the protocol to generate  $N$ , each party  $i$  publishes  $\alpha^{p_i+q_i} \bmod R$ . Then, during the primality test, each party can, if challenged, prove that the value of  $p_i + q_i$  used to calculate  $\alpha^{p_i+q_i} \bmod R$  is the same as the value used to calculate  $x^{p_i+q_i} \bmod N$ . In a similar way, proofs can be given for the values used in the Boneh Franklin primality test.

## 6. Reference

[1] C. Cocks *Split Knowledge Generation of RSA Parameters*. In Cryptography and Coding - Proceedings of 6th IMA International Conference. Lecture Notes in Computer Science vol 1355.

[2] D. Boneh and M. Franklin *Efficient Generation of Shared RSA Keys*. In Proceedings of Crypto '97. Lecture Notes in Computer Science vol 1294.

[3] D. Coppersmith *Private Communication*.

[4] R. T. Moenck Math Comp **31** (1977) pp 235-250

[5] S. Blackburn, S. Blake-Wilson, M. Burmester and S. Galbraith *Secure Construction of Shared RSA Keys*. Paper to appear