

# Split Knowledge Generation of RSA Parameters

Clifford Cocks

## Abstract

We show how 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 instead follow the co-operative procedure described.

## 1. INTRODUCTION

The parameters for the well known RSA system consist of a public modulus  $N$  which is a product of two primes, a public encipherment key  $e$ , and a secret decipherment key  $d$ . The factorisation of  $N$  is a secret parameter and the keys are related by the formula  $de \equiv 1 \pmod{\Phi(N)}$ , where  $\Phi(N)$  is the order of the multiplicative group of integers modulo  $N$ . Then with knowledge of only the public parameters, any message  $x$  (represented as a positive integer less than  $N$ ) can be enciphered as  $y = x^e \pmod{N}$ . The secret parameter  $d$  is needed to decipher the encrypted message  $y$  via the formula  $x = y^d \pmod{N}$ .

In this paper we show how it is possible for two participants to co-operate in generating the public parameters  $N$  and  $e$ , in such a way that individually neither knows the factorisation of  $N$ , and such that they each have a share  $d_1$ ,  $d_2$  respectively of the secret decrypt exponent  $d$  where  $d = d_1 + d_2$ . Hence neither participant has the ability to recover  $x$  from an enciphered message  $y$ , (where  $y = x^e \pmod{N}$ ), but as we show they can enable recovery of  $x$  if they jointly agree and follow a specific decryption procedure. This procedure does not compromise knowledge of the secret parameters, and can be executed in such a way that a third party is involved, and that party alone is able to recover the value of  $x$ .

Boneh and Franklin in independent work [1] describe a method that is similar to ours. However, it differs principally in that they require the help of a third party to generate the RSA parameters. Our approach avoids the need for this - at the cost of increasing the amount of computation required.

## 2. APPLICATIONS

One possible application of this method is in split escrow schemes, where a user will deposit an encryption key with two escrow agents, for possible retrieval later by a duly authorised entity [2]. The protocols described in this paper can be used to ensure that neither escrow agent acting alone has access to the encryption key, but that recovery is possible by co-operative action. Furthermore the escrow

agents do not themselves need to gain access to the deposited key even when executing the recovery procedure.

Another application is to the Fiat Shamir signature scheme [3] which requires a trusted centre to issue secret identification data to new users on registration. This centre necessarily has all the information needed to allow it to masquerade as any registered user. We show how the parameter generation method described in this paper allows for two centres to split the information so that neither of them individually can masquerade as a user or forge their signature.

### 3. A SIMPLE METHOD

Before describing our method it is worth noting that there is a simple extension of the RSA system that can provide some of the desired functionality. In the case of a split escrow scheme suppose that the two agents separately generate moduli  $N_1$  and  $N_2$  and agree on a common public exponent  $e$ . The corresponding secret exponents will be  $d_1$  and  $d_2$ , each known by only one agent. Then they make known the public modulus  $N = N_1N_2$ . Now given an enciphered message  $y$  where  $y = x^e \pmod N$ , each agent is only able to recover  $x \pmod{N_1}$  and  $x \pmod{N_2}$  respectively, but if they share these then together they can easily recover  $x$ .

This approach has undesirable features which we avoid with the more complex protocol described in the paper. The first is that the length of the public modulus  $N$  will need to be twice as long as for a normal RSA system and this will make the system slower at the user level.

The second undesirable feature is the fact that each agent acting alone can recover one of  $x \pmod{N_1}$  and  $x \pmod{N_2}$ . This fact places strong constraints on the way that  $x$  would need to be encoded to ensure that this information is of no value.

### 4. THE NEW METHOD - OVERVIEW

The method proposed consists of three parts:

- a. A procedure to enable two agents to generate a modulus  $N$  that is (to a high probability) the product of two primes  $P$  and  $Q$ , but neither agent can recover  $P$  or  $Q$  with a feasible amount of work.
- b. A procedure for the two agents to generate their shares  $d_1$  and  $d_2$  of the secret recovery exponent  $d$ , given a public encryption exponent  $e$ .
- c. A procedure to allow cooperative recovery of an encrypted message.

## 5. GENERATION OF MODULUS

In this section I will refer to the two cooperating agents as Alice and Bob. Alice will generate two numbers  $p_1$  and  $q_1$  (not necessarily prime) and Bob will generate  $p_2$  and  $q_2$ . We first show how they can generate the number  $N = PQ$  where  $P = p_1 + p_2$  and  $Q = q_1 + q_2$  in such a way that neither knows  $P$  or  $Q$ , and then show how they can test  $N$  for the condition that  $P$  and  $Q$  are probably prime. This procedure is repeated until they find an  $N$  for which the test is satisfied, and this  $N$  becomes the public modulus.

a. To begin with Alice chooses her own RSA modulus  $M$  and public exponent  $e_M$ , which she makes known to Bob. Alice's secret decrypt exponent will be  $d_M$ . The size of  $M$  must be at least as big as the largest size of the public modulus  $N$  they wish to generate. Then Alice sends to Bob the quantities  $p_1^{e_M} \bmod M$  and  $q_1^{e_M} \bmod M$ .

b. Bob can now calculate the three numbers  $(p_1q_2)^{e_M} \bmod M$ ,  $(p_2q_1)^{e_M} \bmod M$  and  $(p_2q_2)^{e_M} \bmod M$ . We will call these three quantities  $a_1$ ,  $a_2$  and  $a_3$ .

c. Bob also generates a set of numbers  $b_{i,j}$  for  $i = 1, 2, 3$  and  $j=1, 2, \dots, K$ . The value of  $K$  will be discussed later in section 10 (Security Issues). The  $b_{i,j}$  are chosen to be random modulo  $M$ , subject to the constraints:  $\sum_j b_{1,j} = \sum_j b_{2,j} = \sum_j b_{3,j} = 1$

d. Bob then generates the  $3K$  numbers  $x_{i,j} = a_i b_{i,j}^{e_M} \bmod M$  and sends these to Alice **but in a new order**. The ordering must be such that it is computationally infeasible to recover the values of  $i$  and  $j$  from the set  $x_{i,j}$  as sent. Thus a random order, or a sorted order are both acceptable.

e. Alice can now calculate  $y_{i,j} = x_{i,j}^{d_M} \bmod M$ . Thus  $y_{i,j} = b_{i,j} a_i^{d_M} \bmod M$ , and hence Alice can determine

$$N = (p_1 + p_2)(q_1 + q_2) = p_1q_1 + \sum_{i,j} y_{i,j} \bmod M, \text{ which she sends to Bob.}$$

As it stands Alice could cheat by substituting any  $N$  of her choice at this point (although if she does, it is not clear that she will be able to complete the later steps of the method in a satisfactory fashion). Nevertheless, we can prevent such cheating by making the above procedure symmetrical. To do this Bob produces his own RSA modulus and executes the above exchange using the same values of  $p_1$ ,  $p_2$ ,  $q_1$  and  $q_2$ , and for this exchange it will be Alice who produces the set of  $3K$  random values. Then Alice and Bob will both know  $N$ , so they exchange a hash of  $N$  to confirm that they have recovered the same value.

## 6. PRIMALITY TESTING

There is no particular reason to suppose that the  $P$  and  $Q$  produced in this way are prime, so it will be necessary for the above procedure to be carried out many times until an  $N$  is found that is likely to be the product of two primes

and can be used as an RSA modulus. Thus we need a test for  $N$  being of the right form, and we propose to test for the condition that  $x^{N+1} \equiv x^{P+Q} \pmod N$  for many  $x$ . (In practice almost all  $N$  not of the correct form will fail on the first  $x$  tested.) This condition does not guarantee that  $P$  and  $Q$  are prime, for example either of them could be Carmichael numbers. However, even if they are not prime we can use  $N$  as an RSA modulus, but being a product of more than two primes may make such  $N$  easier to factorise.

To test whether  $N$  is of the right form Alice and Bob will agree on a set of  $x$  values to use, and Alice will calculate  $x^{N+1-p_1-q_1} \pmod N$  and Bob will calculate  $x^{p_2+q_2} \pmod N$ . They will exchange a hash (using a secure hash function) of these values and this will be sufficient to tell if they are equal.

It is possible to use the test of Boneh and Franklin [1] to increase confidence in the fact that  $P$  and  $Q$  are both prime, once an  $N$  has been found that passes the above test. To make use of this it is necessary that  $P \equiv 3 \pmod 4$  and  $Q \equiv 3 \pmod 4$ . This can be achieved by agreeing the values of  $p_1$ ,  $p_2$ ,  $q_1$  and  $q_2$  modulo 4 in advance. The test, which is described below for the sake of completeness, will be executed  $k$  times, where  $k$  is set according to the level of confidence required. A number  $N$  of the correct form will always pass the test, whilst if either  $P$  or  $Q$  is composite the test will fail with probability at least  $1/2$  at each of the  $k$  iterations.

Each test consists of two steps. At step 1, Alice and Bob will agree on a random integer  $x$  in the range 1 to  $N - 1$  such that the Jacobi symbol  $\left(\frac{x}{N}\right)$  equals  $+1$ . Alice then calculates a hash of each of the two values  $\pm x^{\frac{(N+1+l-p_1-q_1)}{4}} \pmod N$ , and Bob calculates a hash of  $x^{\frac{(l+p_2+q_2)}{4}} \pmod N$ .  $l$  will be 0,1,2 or 3 according to the agreed values of  $p_1$ ,  $p_2$ ,  $q_1$  and  $q_2$  modulo 4. The hashed values are compared, and there must be a match or the test fails.

At step 2, Alice and Bob agree on two random co-prime integers in the range 1 to  $N - 1$ ,  $u$  and  $v$  say. They work in the ring of polynomials  $Z_N[X]$ , and Alice computes the remainder when  $(uX + v)^{N+1+p_1+q_1}$  is divided by  $X^2 + 1$ , and Bob computes the remainder when  $(uX + v)^{p_2+q_2}$  is divided by  $X^2 + 1$ . Writing these polynomials as  $u_1X + v_1$  and  $u_2X + v_2$  respectively, Alice calculates a hash of  $v_1/u_1 \pmod N$  and Bob calculates a hash of  $-v_2/u_2 \pmod N$ . These are compared and the test fails if the two values differ.

## 7. GENERATION OF EXPONENT

Once an acceptable modulus has been found, the next step is to generate the public exponent  $e$  and the secret exponent  $d$ . This is done in such a way that  $d$  is held by the cooperating partners Alice and Bob in two parts  $d_1$  and  $d_2$ , where  $d_1 + d_2 = d$ . The process of generating these parameters is as follows:

Firstly, the public exponent  $e$  is agreed. This should be chosen so that  $P - 1$  and  $Q - 1$  are likely to be coprime to  $e$ , but at the same time  $e$  should not be too large as Alice and Bob will have to share  $(p_1 + q_1) \pmod{e}$  and  $(p_2 + q_2) \pmod{e}$  with each other. A value such as  $e = 2^{16} + 1$  should be satisfactory.

Now Alice reveals  $(p_1 + q_1) \pmod{e}$  to Bob and Bob reveals  $(p_2 + q_2) \pmod{e}$  to Alice, so they can both calculate  $f = P + Q - N - 1 \pmod{e}$ . If  $f$  is non-zero and co-prime to  $e$  then they can both calculate  $g = f^{-1} \pmod{e}$ . The secret decrypt exponent will be:

$$d = \frac{(1+(N+1-P-Q)g)}{e}, \text{ but}$$

$$\text{Alice will calculate: } d1 = \lfloor \frac{1+(\frac{(N+1)}{2}-p_1-q_1)g}{e} \rfloor$$

$$\text{Bob will calculate: } d2 = \lceil \frac{(\frac{(N+1)}{2}-p_2-q_2)g}{e} \rceil$$

It can be easily verified that  $d = d_1 + d_2$ .

## 8. DATA RECOVERY PROCEDURE

We suppose that a third party, Carol say, is authorised to obtain the decrypt  $x$  from an enciphered message  $y = x^e \pmod{N}$ . Then she presents  $y$  to Alice and Bob and receives back  $x_1 = y^{d_1} \pmod{N}$  and  $x_2 = y^{d_2} \pmod{N}$  respectively. Thus Carol can determine  $x = x_1 x_2 \pmod{N}$ . Obviously, Alice and Bob could recover this information themselves by sharing  $x_1$  and  $x_2$  and then presenting the recovered value of  $x$  to Carol.

## 9. USE WITH FIAT SHAMIR SCHEME

In the case of the Fiat Shamir scheme [3], where Alice and Bob take the place of the single trusted center, Carol will need to present values  $v_j$ , derived via her identity and a universal hash function, and obtain data that will allow her to calculate  $s_j$ , where  $s_j^2 = v_j \pmod{N}$  whenever  $v_j$  has a square root.

To do this with our scheme, we require that  $P \equiv 3 \pmod{4}$  and  $Q \equiv 3 \pmod{4}$ , and note that if  $d = \frac{(P-1)(Q-1)+4}{8}$ , then whenever  $v_j$  is a square modulo  $N$  then  $s_j = v_j^d \pmod{N}$  is a square root. The exponent  $d$  will be held in two parts  $d_1$  and  $d_2$  by Alice and Bob respectively.

$$\text{For Alice: } d1 = \lfloor \frac{(\frac{(N+5)}{2}-p_1-q_1)}{8} \rfloor$$

$$\text{For Bob: } d2 = \lceil \frac{(\frac{(N+5)}{2}-p_2-q_2)}{8} \rceil$$

Then if presented with a value of  $v_j$ , Alice and Bob must check that the Jacobi symbol  $(\frac{v_j}{N})$  equals 1. If so Alice will calculate  $w_{j,1} = v_j^{d_1} \pmod{N}$  and Bob will calculate  $w_{j,2} = v_j^{d_2} \pmod{N}$ . Carol will calculate  $w_j = w_{j,1} w_{j,2} \pmod{N}$ , which will either be the square root of  $v_j \pmod{N}$ , or will be the square root of  $N - v_j \pmod{N}$ . In the latter case Carol will reject this particular  $v_j$ .

## 10. SECURITY ISSUES

A critical question is the size of  $K$ , the number of fragments into which Bob splits each of the three quantities  $(p_1q_2)^{e_M} \bmod M$ ,  $(p_2q_1)^{e_M} \bmod M$  and  $(p_2q_2)^{e_M} \bmod M$ . Alice receives (encrypted under her modulus)  $p_1q_2b_{1,j}$ ,  $p_2q_1b_{2,j}$  and  $p_2q_2b_{3,j}$ . She can recover the factorisation of  $N$  if she can identify which fragment is associated with each of the three quantities  $p_1q_2$ ,  $p_2q_1$  and  $p_2q_2$ .

We propose that  $K$  be chosen so that the total number of possible arrangements  $\frac{(3K)!}{(K!)^3}$ , exceeds  $M^2$ . This ensures that for most guesses by Alice as to the value of  $p_1q_2$ ,  $p_2q_1$  and  $p_2q_2$  (subject to their sum being the value recovered) there will be a partition of the  $3K$  fragments into 3 sets which produce these three values. In other words, Alice gains negligible additional information about the values of  $p_1q_2$ ,  $p_2q_1$  and  $p_2q_2$  from the fact that they can be obtained from a partition of the  $3K$  pieces. If  $M$  is 512 bits in size then  $K$  will need to be at least 218 to achieve this bound, and for 1024 bits  $K$  will need to be at least 433. In practice it is likely that smaller values of  $K$  will provide adequate security, but a detailed analysis is beyond the scope of this paper.

## 11. COMPUTATIONAL ISSUES

Assuming that Alice's public encryption exponent  $e_M$  is small, the principal amount of work is performed by Alice, who calculates  $3K$  decryptions for each trial  $N$ . As the probability that  $N$  is a product of two primes is about  $\frac{1}{(\log P)(\log Q)}$ , the total amount of work Alice will expect to have to do amounts to  $0.75K(\log N)^2$  decryptions, assuming that  $P$  and  $Q$  are of similar size. For numbers of size 512 bits, with  $K$  equal to 218, this is about 20.6 million decryptions. At 1024 bits, with  $K$  equal to 433, the expected number of decryptions is about 164 million. Clearly it is desirable to cut the work down if at all possible.

One way to do this is to increase the probability that  $P$  and  $Q$  will be prime. This can be achieved by taking a set of small primes:  $3, 5, 7, \dots, R$  say and agreeing that both Alice and Bob will choose  $p_i \bmod S$  and  $q_i \bmod S$  to be less than  $\lfloor S/2 \rfloor$  for each prime  $S$  in this range, and also agree that only one of them can choose numbers that are a multiple of  $S$ . For the prime 2, they must agree in advance who will select a multiple of 2 and who will choose an odd number. (If they are going to use the Boneh and Franklin primality test then they will control the values of  $p_i$  and  $q_i$  modulo 4 as well).

Each small prime places 1 bit of constraint for Alice and Bob on the choice of  $p_i$  and  $q_i$ , but significantly reduces the number of candidate moduli  $N$  that need to be tested. An alternative would be to use zero knowledge methods to ensure that  $p_1 \bmod S \neq -p_2 \bmod S$  and that  $q_1 \bmod S \neq -q_2 \bmod S$ , but this will not be necessary if the number of primes to be controlled is small. If the first 10 primes are controlled so that  $R = 29$ , the number of decryptions Alice needs

to make drops by a factor of 40. Using the first 20 primes drops the expected number by a factor of about 60. This helps to make the work involved practical, at least for generating long term system wide RSA parameters. For example, if the first 20 primes are controlled, the calculations needed to generate a 512 bit modulus would take a little over one day to complete using MATHEMATICA on a SPARC10 workstation.

## 12. Acknowledgement

I thank my colleague Richard Smith for valuable discussions on this work.

## 13. References

- [1] D. Boneh and M. Franklin *Efficient Generation of Shared RSA Keys*. Submitted to Crypto 97.
- [2] S. Micali *Fair Cryptosystems*. MIT Technical report MIT/LCS/TR-579.h, November 1993
- [3] A. Fiat and A. Shamir *How to prove yourself: Practical solutions to identification and signature problems*. In *Advances in Cryptology - Crypto 86* Lecture Notes in Computer Science vol 263, pp 186-194