Published in D. Augot and C. Carlet, Eds, *Workshop on Coding and Cryptography*, pp. 157–163, Paris, January 11–14, 1999.

Cheating in split-knowledge RSA parameter generation

Marc Joye[1] and Richard Pinch[2]

[1] Laboratory of Cryptography and Information Security,
Dept of Electrical Engineering, Tamkang University
Tamsui, Taipei Hsien, Taiwan 25137, R.O.C.
Email: joye@ee.tku.edu.tw
[2] Queens' College
Silver Street, Cambridge CB3 9ET, U.K.
Email: rgep@cam.ac.uk

Abstract. Cocks [4] has recently described a protocol for two parties to generate an RSA modulus $N = PQ$ where neither party has knowledge of the factorisation, but which enables the parties to collaborate to decipher a encrypted message. We describe a number of ways in which it is possible for one of the parties to the protocol to cheat and obtain knowledge of the factorisation, and suggest modifications to the protocol to guard against cheating.

1 Introduction

Cocks [4] has recently described a protocol for two parties to generate an RSA modulus $N = PQ$ where neither party has knowledge of the factorisation, but which enables the parties to collaborate to decipher a encrypted message. An alternative method is described by Boneh and Franklin [2].

His protocol allows two parties A and B to form an RSA modulus $N = PQ$ in such a way that neither party has knowledge of the factorisation, but the two parties can combine later to decipher a encrypted message. In the proposal of Boneh and Franklin, A and B need the services of a trusted helper H to carry out the distributed computation to form the modulus N: the proposal of Cocks requires no additional parties.

We show that there are a number of ways in which the parties to Cocks's protocol can cheat: that is, obtain knowledge of the factorisation of N or force their own choice of N. In each case the cheating party would later be able to decrypt messages without the collaboration of the other party.

2 The protocol of Cocks

We briefly describe the proposed protocol. There are three stages: generation of N; testing that N is a product of two primes; sharing the encryption and decryption exponents. The protocol of Boneh and Franklin differs in the first stage, where the trusted helper H is employed: the remaining two stages are broadly similar.

At the beginning of the first stage, A secretly chooses two random numbers p_1 and q_1 and B similarly chooses p_2 and q_2. Their intention is to form $N = PQ$ where $P = p_1 + p_2$ and $Q = q_1 + q_2$, and P and Q are prime. We assume that A and B have their own RSA moduli N_A and N_B with enciphering exponents e_A and e_B and deciphering exponents d_A and d_B.

In the first stage A transmits $p_1^{e_A}$ and $q_1^{e_A}$ to B. B now chooses three sets of k numbers $x_{1,i}$, $x_{2,i}$, $x_{3,i}$, such that for each j we have $\sum_i x_{j,i} = 1$. He then forms the set of $3k$ numbers

$$X = \left\{ x_{1,i}^{e_A} p_1^{e_A} q_2^{e_A}, x_{2,i}^{e_A} p_2^{e_A} q_1^{e_A}, x_{3,i}^{e_A} p_2^{e_A} q_2^{e_A} : i = 1, \ldots, k \right\}$$

and transmits the elements of X in some order to A. We call this the "splitting process" and say that B has split the values p_1q_2, p_2q_1 and p_2q_2. A now forms

$$N = p_1q_1 + \sum_{x \in X} x^{d_A} = p_1q_1 + p_1q_2 + p_2q_1 + p_2q_2 = PQ$$

and transmits the result to B. It is also proposed that A and B might wish to carry out this stage with rôles reversed: we refer to this as the "symmetric" case.

In the second stage A and B agree a random value b. A forms $b^{N+1-p_1-q_1} \bmod N$ and B forms $b^{q_1+q_2} \bmod N$. If these disgree then $\phi(N)$ cannot equal $N+1-P-Q = (P-1)(Q-1)$ and so P and Q cannot both be prime. A and B exchange hashes of these values and if they disagree then N is rejected and the protocol restarts at the first stage. If sufficiently many trial b pass this test, then P and Q are declared (probably) prime.

Boneh and Franklin suggest a modification of this test to avoid the possibility that P or Q might be a Carmichael number (which would pass the proposed test unless b actually has a factor in common with N). They propose that P and Q should be chosen to be $\equiv 3 \bmod 4$ by suitable conditions on the p_i and q_i. Then b is chosen so that the Jacobi symbol $\left(\frac{b}{N}\right) = +1$ and the necessary condition $b^{(N-1)/4} \equiv \pm 1 \bmod N$ is tested by comparing $\pm b^{(N-1-p_1-q_1)/4}$ and $b^{(p_2+q_2)/4}$. Again if these do not agree then N is rejected, otherwise if N passes sufficiently many of these tests then it is allowed to proceed to the next stage. The advantage of this refinement is that if P and Q are not both prime then the probability of the test succeeding with a randomly chosen b is less than $1/2$.

In the third stage an enciphering exponent e is agreed on (perhaps fixed in advance). A and B exchange the values $p_i + q_i \bmod e$. Each can now compute $f = P + Q - N - 1 \bmod e$ and its inverse $g = f^{-1} \bmod N$. A forms

$$d_1 = \left\lfloor \frac{1 + \left(\frac{N+1}{2} - p_1 - q_1\right)g}{e} \right\rfloor$$

and B forms

$$d_2 = \left\lceil \frac{\left(\frac{N+1}{2} - p_2 - q_2\right)g}{e} \right\rceil$$

Since

$$d = d_1 + d_2 = \frac{(N + 1 - P - Q)g}{e},$$

we have $de \equiv 1 \bmod \phi(N)$. Hence to decrypt $C = M^e \bmod N$, A can form C^{d_1} and B can form C^{d_2}, so that $M = C^{d_1} \cdot C^{d_2}$.

3 Preliminary remarks

From the discussion of the protocol, it is clear that either A or B can cheat if they can obtain the secrets p_i and q_i of the other party. With this information the cheater can mimic the computations of the other party in order to recover the deciphering exponent d, or predict the other's responses in order to impose a different value N' and have it accepted as the correct modulus.

In particular, if either party can obtain both the values of $p_i + q_i$ at any point, then from that point on they can privately simulate the other party's computations. This will enable the cheater to ensure that any given N (whether or not the result of carrying out that stage correctly) is passed as being of the appropriate form, and also allow the cheater to participate in joint deciphering, without the cheating being detected.

We further note that the protocol may be expected to be repeated a large number of times (Cocks [4] suggests 10^4 times for a modulus of 1024 bits). A cheater can therefore afford to wait for a reasonable amount of good luck, deliberately causing the protocol to repeat until it occurs.

4 Cheating in the first stage

Clearly A can cheat in the first stage by announcing any value N' she please as the result of the computation. However, in order to have any chance of passing the second stage, it seems that A needs to obtain p_2 and q_2 in order to predict B's contribution $b^{p_2+q_2}$. We shall see in the next section that this is not the case. However, a possible way to obtain p_2, q_2 is for A to privately carry out the computation correctly in order to obtain the true value of N. She may then attempt to factor this. If it is possible to factor N quickly, for example if it has only small prime factors and so is easy to factor by trial division, then A may be able to obtain the secrets p_2 and q_2 from the resulting factors.

It is also possible for B to cheat. Suppose that B applies the splitting process to the numbers $(-p_1q_1)^{e_A}$, α^{e_A} and β^{e_A}. We note that B can do this, using $(-p_1q_1)^{e_A} = -p_1^{e_A}q_1^{e_A}$, since e_A must be odd. A will then form $p_1q_1 - p_1q_1 + \alpha + \beta = \alpha + \beta$. So B can force an arbitrary modulus N' by taking α at random and $\beta = N' - \alpha$. However, B will have no more information about p_1 and q_1 than he would have had by applying the protocol correctly (that is, B will possess only $p_1^{e_A}$ and $q_1^{e_A}$).

Here we see that performing the first stage symmetrically actually weakens it by making it possible for B to cheat in the same way as A: since he is playing

the rôle of A in the reversed computation, he can obtain the true value of N privately and has the chance of factoring it.

It is possible for B to cheat in the asymmetric case. He chooses a prime r and applies the previous method with $\alpha = p_1 r$ and $\beta = q_1 r$. A will now form $N' = \alpha + \beta = (p_1 + q_1)r$. A returns N' to B, who knows r and hence obtains $p_1 + q_1$. Although B has not imposed a modulus of his own choice, he now has the factors of the modulus N' agreed on and enough information required to force it through the remaining stages.

5 Cheating in the second stage

Suppose now that one of the parties has imposed a false value N' on the other during the first stage. We have already observed that it is possible to force this number to pass the second stage given information about the $p_i + q_i$.

We now show that it is possible to cheat without this knowledge: for example, if B has cheated in an asymmetric first stage. Suppose that N' has been imposed by the cheater after choosing it so that all its prime factors p satisfy $p - 1 \mid L$ for some L. For example, taking $L = 2^5 3^4 5.7 = 90720$, there are 49 such primes, with a product in excess of 10^{270}. It is certainly possible to form a value of N' of say 512 bits. Then the exponent $\lambda(N')$ of the multiplicative group modulo N' will divide L and there is a chance of at least $1/L$ that the value $b^{N'+1-p_1-q_1}$ or $b^{p_2+q_2}$ formed during the computation will be $\equiv 1 \bmod N'$. So the cheater simply hopes that this will be the case, and tries again if not.

6 Preventing cheating

We note that certain forms of cheating will require the cheater to undertake a certain, possibly considerable, amount of extra computation compared to an honest participant. We suggest that, as a general principle, timing requirements should be included in any such protocol and that a session should be aborted if either party fails to reply in a timely manner.

A related requirement is that the protocol should specify a maximum number of repetitions consistent with the probability of finding a number of the required form. If the cheater is waiting for some other event which is less likely than the legitimate one, this should lead to the protocol being aborted before the cheater succeeds.

We further suggest that at the end of the first stage, either party should have the option of randomly requiring that the round terminate with each side immediately revealing their choice of random p_i, q_i. It would then be open to both parties to verify that the data which has passed is consistent with the revealed secret. A cheater should not be able to do this and so will be detected. As a further security measure we suggest that, as in all similar protocols for generation of secret keys, an audit trail should show the pseudo-random generation from

an initial seed, and that this should be done through a cryptographically strong generator to prevent a 'seed' being determined from the output.

We note that a number of the methods of cheating that we have proposed are in fact easy to detect. For example, one method leads to a modulus N' divisible by $p_1 + q_1$: another leads to N' being divisible only by small primes. One might advocate building checks against these into the protocol. However, we note that they do little to guard against other, as yet undiscovered methods of cheating, and hence may be thought to lead an unwarranted degree of belief in the validity of a run which passes these tests alone.

7 Does cheating matter?

At first sight, it might appear that cheating is not an important issue for the intended application. The parties involved will presumably be acting as "trusted third parties" and it might be held that trusted parties do not cheat. We do not reject this out of hand: but it is perhaps worth noting that the parties involved are likely to be large corporations or government departments, and it is not so clear that every employee or sub-contractor will be equally trustworthy.

We might also note that for political acceptability of such arrangements, it is desirable to minimise the amount of reliance which the general public is required to place in the trustworthiness of these parties. Fielding a protocol which is known to be vulnerable to cheating is likely to diminish public confidence and encourage opposition from those already predisposed to resist the principles involved.

References

1. Simon Blackburn, Simon Blake-Wilson, Mike Burmester, and Steven Galbraith, *Shared generation of shared RSA keys*, Tech. Report CORR 98-19, Dept of C&O, University of Waterloo, Canada, February 1998.
2. Dan Boneh and Matthew Franklin, *Efficient generation of shared RSA keys*, In Kaliski Jr [6], pp. 425–439.
3. Clifford Cocks, *Split generation of RSA parameters with multiple participants*, Available at `http://www.cesg.gov.uk/`.
4. ———, *Split knowledge generation of RSA parameters*, In Darnell [5], pp. 89–95.
5. Michael Darnell (ed.), *Cryptography and Coding*, Lecture Notes in Computer Science, vol. 1355, Springer-Verlag, 1997.
6. Burton S. Kaliski Jr (ed.), *Advances in Cryptology – CRYPTO '97*, Lecture Notes in Computer Science, vol. 1294, Springer-Verlag, 1997.