RSA Weaknesses - Solutions

Solution 1: The idea here is that since the two prime factors of N are close, they must lie in the vicinity of $\sqrt{N}$, which here is roughly 20016. Searching for factors in either direction results in a quick factorisation of $N = 20011 \times 20021$.

The maple problem is solved in the same way but here a manual search is not practical (the primes used are much further apart). The maple code:

N := [enter value of N];
for j **from** 1 **to** 5000 **do** **if** N mod (isqrt(N)+ j) = 0 **then** print(isqrt(N) + j) **fi; od;**

finds a prime divisor of N immediately. There are of course many other ways to write such a program. This extension problem shows that even using big and close primes is still as much a weakness as small and close primes. In practice there are better methods that exploit this weakness such as Fermat factorisation and the Quadratic sieve

Solution 2: (i) We know that $c \equiv p^3 \bmod N$, where p is the required plaintext. But here we are lucky in that c is actually a perfect cube, so that we may get the plaintext by solving the equation $c = p^3$ instead. Hence $p = \sqrt[3]{c} = 762$.

(ii) We notice that for any such plaintext we have that $p^e < N$. But also we have that $c < N$ (by definition). Thus, after encryption we not only have that $c \equiv p^e \bmod N$ but we see that the more stronger relationship, $c = p^e$ holds, giving the plaintext as $p = \sqrt[e]{c}$. This agrees with (i).

Here we see a restriction to the plaintexts that you can send when using RSA (if you want your system to be secure). If p is small enough then after raising to the eth power and reducing mod N we do not really use the "circular" nature of arithmetic mod N and so we haven't really hidden the plaintext value at all.

It is easy to see that this is more of a restriction for small e. For a fixed N, the value of $N^{\frac{1}{e}}$ gets smaller as e gets bigger, meaning there is more flexibility in the plaintexts that we may use. This is certainly a weakness of "low public exponent RSA" but it is not a major one, we can simply choose the way we turn our plaintext message into a number to mostly avoid making such small values.

Solution 3: Let the plaintext be p and the two intercepted ciphertexts be c_1 and c_2. Now the fact that e_1 and e_2 are coprime guarantees the existence of integers a and b such that $ae_1 + be_2 = 1$ (and since the values e_1 and e_2 are known we can find such a, b easily by using Euclid's algorithm).

Once this is done we may find p via:

$$p = p^{ae_1+be_2} = (p^{e_1})^a (p^{e_2})^b \equiv c_1^a c_2^b \bmod N.$$

(Recall that we know the values of c_1 and c_2 through interception).

Solution 4: (i) Suppose, without loss of generality, that $\gcd(N_1, N_2) > 1$. The fact that N_1 and N_2 are distinct semi-primes means that the value of this gcd must be a prime divisor of both N_1 and N_2, so that we have found one of the two prime factors of N_1. We can easily find the other by division and hence break the system by finding this persons private key.

(ii)The following congruences hold:

$$p^3 \equiv c_1 \bmod N_1,$$
$$p^3 \equiv c_2 \bmod N_2,$$
$$p^3 \equiv c_3 \bmod N_3.$$

(iii) Since we are assuming the N_i values to be pairwise coprime we may use the Chinese Remainder Theorem to solve the congruences in (ii) for the unknown p^3. This gives us a unique solution mod $N_1 N_2 N_3$, say:

$$p^3 \equiv n \bmod N_1 N_2 N_3,$$

where $n < N_1 N_2 N_3$. This n is easily calculated.

But actually we have that $p^3 < N_1 N_2 N_3$ too since $p < N_i$ for $i = 1, 2, 3$ (this is because the RSA protocol insists that the plaintext p is always less than the public modulus of the person you wish to communicate with).

This tells us that we really have an equality of integers, $p^3 = n$ and since n can be calculated, we can simply find the plaintext p via $p = \sqrt[3]{n}$.

The extension is solved in the same way. We may again assume that $\gcd(N_i, N_j) = 1$ for all distinct $1 \leq i, j \leq e$, or else we may use Euclid's algorithm to find the value of a "non-1" gcd, this being a prime divisor of one of the moduli (allowing us to factorise one of the moduli and break the system).

Making the assumption above, we now have the e congruences:

$$p^e \equiv c_1 \bmod N_1,$$
$$p^e \equiv c_2 \bmod N_2,$$
$$.$$
$$.$$
$$.$$
$$p^e \equiv c_e \bmod N_e.$$

Under our assumption that the moduli are coprime we may now use the Chinese Remainder Theorem to solve for the unknown p^e. We get a unique solution mod $\prod_{k=1}^{e} N_k$, say:

$$p^e \equiv n \bmod \prod_{k=1}^{e} N_k,$$

for some $n < \prod_{k=1}^{e} N_k$. This n is easily calculated.

But we have that $p^e < \prod_{k=1}^{e} N_k$ too, since $p < N_i$ for each i (again, this is due to the RSA protocol). Thus we really must have that $p^e = n$, giving $p = \sqrt[e]{n}$ as the plaintext.

(This attack can apply more generally to situations where the plaintext messages being sent to each person are not the same but are related in some linear fashion.)