A variant of Wiener’s attack on RSA with small secret exponent

Andrej Dujella

Department of Mathematics, University of Zagreb
Bijenicka cesta 30, 10000 Zagreb, CROATIA
E-mail: duje@math.hr
URL: http://web.math.hr/~duje/

To speed up the RSA decryption one may try to use small secret decryption exponent $d$. The choice of a small $d$ is especially interesting when there is a large difference in computing power between two communicating devices. However, in 1990, Wiener showed that if $d < n^{0.25}$, where $n = pq$ is the modulus of the cryptosystem, then there exist a polynomial time attack on the RSA. He showed that $d$ is the denominator of some convergent $p_m/q_m$ of the continued fraction expansion of $e/n$, and therefore $d$ can be computed efficiently from the public key $(n, e)$.

In 1997, Verheul and van Tilborg proposed an extension of Wiener’s attack that allows the RSA cryptosystem to be broken when $d$ is a few bits longer than $n^{0.25}$. For $d > n^{0.25}$ their attack needs to do an exhaustive search for about $2t+8$ bits (under reasonable assumptions on involved partial convergents), where $t = \log_2(d/n^{0.25})$. In 2004, we introduced a slight modification of the Verheul and van Tilborg attack, based on Worley’s result on Diophantine approximations of the form $|\alpha - p/q| < c/q^2$, for a positive real number $c$.

In both mentioned extensions of Wiener’s attack, the candidates for the secret exponent are of the form $d = rq_{m+1} + sq_m$. We test all possibilities for $d$, and number of possibilities is roughly $(\text{number of possibilities for } r) \times (\text{number of possibilities for } s)$, which is $O(D^2)$, where $d = Dn^{1/4}$. There are two principal methods for testing:

1) compute $p$ and $q$ assuming $d$ is correct guess;

2) test the congruence $(M^e)^d \equiv M \pmod{n}$, say for $M = 2$.

Here we present a new idea, which is to apply “meet-in-the-middle” to this second test. Let $2^{sq_{m+1}} \mod{n} = a$, $(2^{sq_m})^{-1} \mod{n} = b$. Then we test the congruence $a^r \equiv 2b^s \pmod{n}$. We can do it by computing $a^r \mod{n}$ for all $r$, sorting the list of results, and then computing $2b^s \mod{n}$ for each $s$ one at a time, and checking if the result appears in the sorted list. This decrease the time complexity of testings phase to $O(D \log D)$ (with the space complexity $O(D)$).

We present also some variants of the proposed attack, which might be relevant for its practical implementation.
References


