Now, we search for the private exponent d among numbers of the forms $26r + 19s$ or $487s - 26t$ or $4896r' + 487s'$. By applying the criterion for testing candidates for $\frac{k}{d}$ described in the proof of Theorem 9.1, we find that $d = 5936963$, which is obtained for $s = 12195$, $t = 77$. $\diamondsuit$

The time complexity of this attack can be improved by applying the method "meet in the middle" for testing candidates, which is demonstrated in [123] and [218, Chapter 5.1.1]. We want to test whether

$$2^{e(rq_{m+1}+sq_m)} \equiv 2 \pmod{n},$$

holds. Let us notice that the index m is almost fixed. Namely, if m' is the largest odd positive integer such that

$$\frac{p_{m'}}{q_{m'}} > \frac{e}{n} + \frac{2.122e}{n\sqrt{n}},$$

then $m \in \{m', m'+1, m'+2\}$.

Let us introduce the notation:

$$2^{eq_{m+1}} \bmod n = a, \quad (2^{eq_m})^{-1} \bmod n = b.$$

Then we actually test the congruence

$$a^r \equiv 2b^s \pmod{n}.$$

We can do that by calculating $a^r \bmod n$ for all r; we sort the results and then calculate $2b^s \bmod n$ for each s and check whether the result appears in the previously obtained sorted list. In this manner, the number of steps in testing becomes roughly (the number of possibilities for r) + (the number of possibilities for s). To be more precise, the time complexity of the testing phase is reduced from $O(D^2)$ to $O(D \ln D)$ (with the space (memory) complexity $O(D)$). This attack works efficiently for values D less than 2^{30}, i.e. for $d < 2^{30} n^{0.25}$.

9.4 Attacks on RSA using the LLL algorithm

There are also attacks on RSA, which, instead of using continued fractions, use Coppersmith's method for finding solutions to polynomial congruences. Namely, the following problem is encountered. Consider a polynomial $f(x) \in \mathbb{Z}[x]$ of degree d and assume that it is known that there exists a "small" solution of the congruence $f(x) \equiv 0 \pmod{N}$, i.e. the

solution x_0 such that $|x_0| < N^{1/d}$. The question is whether we can efficiently determine x_0. Coppersmith [89] showed that the answer to this question is affirmative. The basic idea is to construct a new polynomial $h(x) = h_0 + h_1 x + \cdots + h_n x^n \in \mathbb{Z}[x]$ such that $h(x_0) \equiv 0 \pmod{N}$, but which will have small coefficients. To be more precise, it is required that the norm $\|h(x)\| := \left(\sum_{i=0}^{n} h_i^2\right)^{1/2}$ is small. Then the following simple fact can be used: if for a positive integer X we have

$$\|h(xX)\| < \frac{N}{\sqrt{n+1}}$$

and $|x_0| < X$ satisfies the congruence $h(x_0) \equiv 0 \pmod{N}$, then x_0 is a root of polynomial h, i.e. it is not only the congruence that is satisfied but also the equation $h(x_0) = 0$.

The polynomial $h(x)$ with the required property can be found by the LLL algorithm. Namely, coefficients of the polynomial $h(x)$ can be obtained as components of the first vector of an LLL-reduced base of a certain lattice which is obtained from coefficients of the initial polynomial $f(x)$.

In 1998, Boneh and Durfee described an attack of this kind which can be applied in the case that $d < n^{0.292}$. Similarly to Wiener's attack, it begins with the equation $ed - k\varphi(n) = 1$, which can be written as

$$ed - k(n + 1 - p - q) = 1.$$

Put $s = p + q$, $a = n + 1$. Now, finding the small private exponent d, say $d < n^{\delta}$, is equivalent to finding the small solutions k and s of the congruence

$$f(k, s) = k(s - a) \equiv 1 \pmod{e}.$$

Indeed, for k and s we have the following estimates:

$$|s| < 3\sqrt{n} \approx e^{0.5}, \quad |k| < \frac{de}{\varphi(n)} \approx e^{\delta}$$

(if d is small, then we expect that e will be large, i.e. $e \approx \varphi(n) \approx n$). Hence, the situation is similar to the above stated result of Coppersmith, except that here we are dealing with a polynomial of two variables so Coppersmith's theorem cannot be directly applied to prove the correctness of this attack rigorously. Still, it was shown that it works satisfactory in practice.

The advice is to avoid the case $d < \sqrt{n}$, because it is known that all above-mentioned attacks are not applicable if $d > \sqrt{n}$. We conclude that

the idea of using a small exponent d in order to accelerate decryption is not a good one.

We will now demonstrate another idea for accelerating decryption, which was proposed by Quisquater and Couvreur in 1982. The standard way of decryption $x = y^d \bmod n$ disregards the fact that (unlike a user who encrypts the plaintext) a user, who is decrypting a message which is intended for him, knows prime factors p and q of the modulus n. Hence, the user can first "partially" decrypt ciphertext modulo p and modulo q by calculating

$$x_p = y^d \bmod p, \quad x_q = y^d \bmod q, \tag{9.3}$$

and then combine those two results by the Chinese remainder theorem to obtain the plaintext modulo n.

The process of decryption can be accelerated if we notice that in (9.3), instead of using the exponent d, we could use exponents d_p and d_q defined by

$$d_p = d \bmod (p-1), \quad d_q = d \bmod (q-1).$$

Namely, from Fermat's little theorem, it follows that

$$x_p = y^{d_p} \bmod p, \quad x_q = y^{d_q} \bmod q.$$

Exponents d_p and d_q are called CRT-exponents. We saw that the exponent for decryption d should not be less than $n^{0.292}$. On the other hand, it seems that there are no obstacles for CRT-exponents to be significantly smaller without putting the security of the cryptosystem in danger. Therefore, in situations when it is desired to minimize the cost of decryption, it is recommended to use this version of RSA. This version is called the *rebalanced RSA*, and with it, first, the small CRT exponents d_p and d_q are chosen, and then, by the Chinese remainder theorem, the corresponding encryption exponent e is calculated.

There are also attacks on RSA under the assumption that the exponent e is small. In early implementations of the RSA cryptosystem, it was often taken $e = 3$ to minimize the time needed for encryption. We will demonstrate the reasons why such a choice for e is not good.

Let us assume that we have three users with distinct values of public moduli n_1, n_2, n_3 and assume that all of them use the same exponent $e = 3$. Furthermore, assume that someone wants to send them an identical message m. Then their opponent can find out the following ciphertexts:

$$c_1 \equiv m^3 \pmod{n_1}, \quad c_2 \equiv m^3 \pmod{n_2}, \quad c_3 \equiv m^3 \pmod{n_3}.$$

After that, he can, by the Chinese remainder theorem, find the solution of the system of linear congruences

$$x \equiv c_1 \pmod{n_1}, \quad x \equiv c_2 \pmod{n_2}, \quad x \equiv c_3 \pmod{n_3}.$$

In this manner, the number $x < n_1 n_2 n_3$ such that $x \equiv m^3 \pmod{n_1 n_2 n_3}$ will be obtained. Since $m^3 < n_1 n_2 n_3$, the equality $x = m^3$ actually holds and the opponent can calculate the original message m by finding the third root of x.

The attack just described can be avoided by adding a "random pad" to the messages before encryption. In this manner, we will never send an identical message to different recipients. However, there are attacks (based on the above-mentioned result of Coppersmith and the LLL algorithm) which show that, in that case, the RSA cryptosystem with a very small exponent e is also not secure. We will now describe Håstad's attack from 1985.

Let us assume that, before encryption, at the beginning of every message some user-dependent information is added. For example,

$$c_i = (i \cdot 2^h + m)^e \bmod n_i, \quad i = 1, \ldots, k.$$

Hence, we have k polynomials $g_i(x) = (i \cdot 2^h + x)^e - c_i$, and we are searching for m such that

$$g_i(m) \equiv 0 \pmod{n_i}.$$

Let $n = n_1 n_2 \cdot \ldots \cdot n_k$. By the Chinese remainder theorem, we can find t_i such that

$$g(x) = \sum_{i=1}^{k} t_i g_i(x) \quad \text{and} \quad g(m) \equiv 0 \pmod{n}$$

(t_i is the solution of the system of congruences $t_i \equiv 1 \pmod{n_i}$, $t_i \equiv 0 \pmod{n_j}$ for $j \neq i$). The polynomial g is monic and of the degree e. If $k \geq e$, i.e. we have at least that many users (interfered ciphertexts) as the value of the public exponent, then $m < \min_i n_i < n^{1/k} \leq n^{1/e}$, so m can be efficiently found by applying the above-mentioned Coppersmith's result.

A detailed overview of different attacks on the RSA cryptosystem and its variants (for example those in which modulus n is the product of more than two prime factors) can be found in the book [218].

The use of exponent $e = 65537$, which is large enough to prevent all known attacks on RSA with a small exponent, can be recommended. Its advantage is very fast encryption because it has a small number of ones in the binary notation. Namely, $65537 = 2^{16} + 1$.