THE DESCRIPTION OF THE PROGRAM FOR FACTORIZATION OF LARGE NUMBERS

Vladimír Levek

Doctoral Degree Programme (5), FEEC BUT

E-mail: levek@feec.vutbr.cz

Supervised by: Pavel Šteffan

E-mail: steffan@feec.vutbr.cz

Abstract: The article describes the algorithm for factorization of large numbers. If there is the result of the product of two prime numbers, then the program can find the factors. The first part of the article generally introduces the problem of factoring large integers and its impact in the field of the cryptography. The next part describes the algorithm and the program for calculation. At the end of the article there is a summary of the possibilities of the program.

Keywords: Cryptography, Factorization, RSA, Prime number, Large Numbers, Order of magnitude

1. INTRODUCTION

The resistance of cryptography, especially asymmetric cryptosystems IF is based on the inability to decompose a large number into the product of smaller integers – factors for the reasonable time. Large number is a number in order of magnitude 10^{300} and higher and the "reasonable time" could be within several months. This task is currently considered insoluble, therefore asymmetric cryptography belongs to the category of highly resistant cryptosystems.

2. ASYMMETRIC CRYPTOSYSTEM RSA

Like any other asymmetric cryptosystems, RSA is primarily designated for encryption of short messages. It is given by computational demands of this system resulting into practical impossibility of the decryption in a short time. The message intended for the encryption must be shorter than numbers used for its security and decryption. The basic mathematical operations for encryption and decryption of messages are

$$C = Z^{VK} \bmod n, \tag{1}$$

$$Z = C^{SK} \bmod n, \tag{2}$$

where C is a message Z encrypted with a public key (VK, n). The message is decrypted with a private key SK. The condition: message Z must be shorter than n, which equals the product of p and q:

$$n = p \cdot q, \tag{3}$$

p a q are prime numbers larger than 10^{150}. These numbers are chosen in such a way that the conditions given below are valid. These numbers also serve for the calculation of modulo r (4) – which is used for the calculation of the private key SK (5). [2][3].

$$r = (p-1)(q-1) \tag{4}$$

$$(VK \cdot SK) \bmod r = 1. \tag{5}$$

Based on the relations above, there is a possibility to get the private key with factorization of the formula (3). As stated above p a q are large prime numbers, therefore it is obvious that the result n is final and therefore factorization will not have more solutions. If cybernetic attacker manages to express p and q from n, he could get r (4) and subsequently he could calculate the private key with the use of condition in (5).

3. FACTORING ALGORITHMS

Algorithm for the calculation of the factors results from eventual compilation of all the possibilities from the lowest order of magnitudes. In each order of magnitude beginning with the lowest, there is a calculation of possible option of factors of the proper order of magnitude. Previous results could be confirmed or disproved. General product A x B can be decomposed to elementary order of magnitudes with the use of modular mathematics.

The product is:

$$A \cdot B = \left(a_0.10^0 + a_1.10^1 + a_2.10^2 ... + a_n.10^n\right) \cdot \left(b_0.10^0 + b_1.10^1 + b_2.10^2 ... + b_n.10^n\right), \quad (6)$$

where A and B are factors and a_x and b_x are integer from 0 - 9. The decomposition of the formula (6) makes a mathematical equation:

$$A \cdot B = a_0 b_0.10^0 + \left(a_0 b_1 + a_1 b_0\right)10^1 + \left(a_0 b_2 + a_1 b_1 + a_2 b_0\right)10^2 ... + \left(a_0 b_n + a_1 b_{n-1} ... + a_{n-1} b_1 + a_n b_0\right)10^n . \quad (7)$$

The formula 7 shows the way the result in a particular order of magnitude is made. For example digit in the result in the second order of magnitude is created in this way:

$$c_2.10^2 = \left(a_0 b_2 + a_1 b_1 + a_2 b_0\right).10^2 . \quad (8)$$

Regarding the fact that the final digit must be between 0-9, the only result would be in the unit order of magnitudes. The whole result is made by modulo of tens. During this process the transmissions to the higher orders of magnitude should be applied. Generally, digits in a particular order of magnitude are made by these formulas:

$$\begin{aligned}
c_0 &= \left(a_0 b_0\right) \bmod 10 \\
c_1 &= \left(a_0 b_1 + a_1 b_0 + P_0\right) \bmod 10 \\
c_2 &= \left(a_0 b_2 + a_1 b_1 + a_2 b_0 + P_1\right) \bmod 10 \\
c_3 &= \left(a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0 + P_2\right) \bmod 10 \\
c_4 &= \left(a_1 b_3 + a_2 b_2 + a_3 b_1 + P_3\right) \bmod 10 \\
c_5 &= \left(a_2 b_3 + a_3 b_2 + P_4\right) \bmod 10 \\
c_6 &= \left(a_3 b_3 + P_5\right) \bmod 10 \\
c_7 &= P_6 \\
c_8 &= 0
\end{aligned} \quad (9)$$

$$\begin{aligned}
P_0 &= \frac{a_0 b_{0_1} - \left(a_0 b_{0_1}\right) \bmod 10}{10} \\
P_1 &= \frac{\left(a_0 b_1 + a_1 b_0 + P_0\right) - \left(a_0 b_1 + a_1 b_0 + P_0\right) \bmod 10}{10} \\
P_2 &= \frac{\left(a_0 b_2 + a_1 b_1 + a_2 b_0 + P_1\right)}{10} \\
&\quad - \frac{\left(a_0 b_2 + a_1 b_1 + a_2 b_0 + P_1\right) \bmod 10}{10} \\
P_3 &= \frac{\left(a_0 b_3 + a_1 b_2 ... + a_3 b_0 + P_2\right)}{10} \\
&\quad - \frac{\left(a_0 b_3 + a_1 b_2 ... + a_3 b_0 + P_2\right) \bmod 10}{10} \\
P_4 &= \frac{\left(a_1 b_3 + a_2 b_2 + a_3 b_1 + P_3\right)}{10} \\
&\quad - \frac{\left(a_1 b_3 + a_2 b_2 + a_3 b_1 + P_3\right) \bmod 10}{10} \\
P_5 &= \frac{\left(a_2 b_3 + a_3 b_2 + P_4\right) - \left(a_2 b_3 + a_3 b_2 + P_4\right) \bmod 10}{10} \\
P_6 &= \frac{\left(a_3 b_3 + P_5\right) - \left(a_3 b_3 + P_5\right) \bmod 10}{10}
\end{aligned} \quad (10)$$

where *c* are final digits as results of modulo and transmissions *P* are defined by the overall result, which is reduced by modulo ten.

4. THE DECRYPTION OF THE PROGRAM

The program for the decomposition of the product was made for calculation of factorization. For the simplification of the whole problematics, it is important to realize that the product of prime numbers will end with digits: 1, 3, 7 or 9. Therefore, the result cannot be even and it is highly probable that it will not end with digit 5.

Number 5 is the only prime number which ends with digit 5. It is supposed that the factors are larger than single-digit. Program assumes that the order of magnitude of the factors is at least two times lower than the order of magnitude of the result.

The programs involves the following steps:

1. Defining the length of factors.

2. Estimation of zero order of magnitude of factors.

3. Calculation of the "corpus" of the order of magnitudes (factors of previous order of magnitudes).

4. Estimation of factors (factors of current order of magnitude).

5. Testing if assumption corresponds with the final calculation

6. If the condition in 5 is fulfilled, the order of magnitude is increased and program continues from the point 2

7. When the calculation in the highest order of magnitude is finished, program finishes, too.

In the first part of the program, the length of factors is calculated. The next part of the program consists of algorithm which determines digits that could be used in zero order of magnitude. "Factor" covers digits of factors in particular order of magnitude – a_x, b_x. The program subsequently tests in dynamic loop all possible variants in particular orders of magnitude until the result is attained. If the result is not attained, the examined number is prime number.

General calculation of each digit in particular order of magnitude is:

$$c_n = \left(a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \ldots + a_n b_0 + P_{n-1}\right) \bmod 10 \,. \tag{11}$$

Because the process of calculation proceeds from the lower orders of magnitude to higher, it is obvious that in 11 there are two unknown variables - a_n a b_n. Other numbers are calculated previously in lower orders of magnitude. Finally, the calculation of the orders of magnitude is divided into a part of "estimation of factors" and a part dealing with calculation of "corpus of the order of magnitude".

$$C_n = \begin{pmatrix} a_0 b_n + a_n b_0 & + \\ a_1 b_{n-1} + a_2 b_{n-2} + a_3 b_{n-3} \ldots + a_n b_0 & + \\ P_{n-1} & \end{pmatrix} \bmod 10 \,. \tag{12}$$

In formula (12) there are three important points:

- Estimation of factors
- Calculation of "corpus of the order of magnitude"
- Addition of the transmission from the lower order of magnitude

While the calculation of "the corpus" is determined by the sum of permutated factor of lower orders of magnitude, estimation of factors is determined by generating factors of current order of magnitude and their multiplying with zero factors.

$$a_0 b_n + a_n b_0 . \tag{13}$$

The relation (13) shows a calculation of newly generated digits, which are generated from 0. In each step it is tested whether the condition in (13) and (14) is valid.

$$a_1 b_{n-1} + a_2 b_{n-2} + a_3 b_{n-3} ... + a_n b_0 , \tag{14}$$

In the figure 1, there is a flowchart of the whole program.

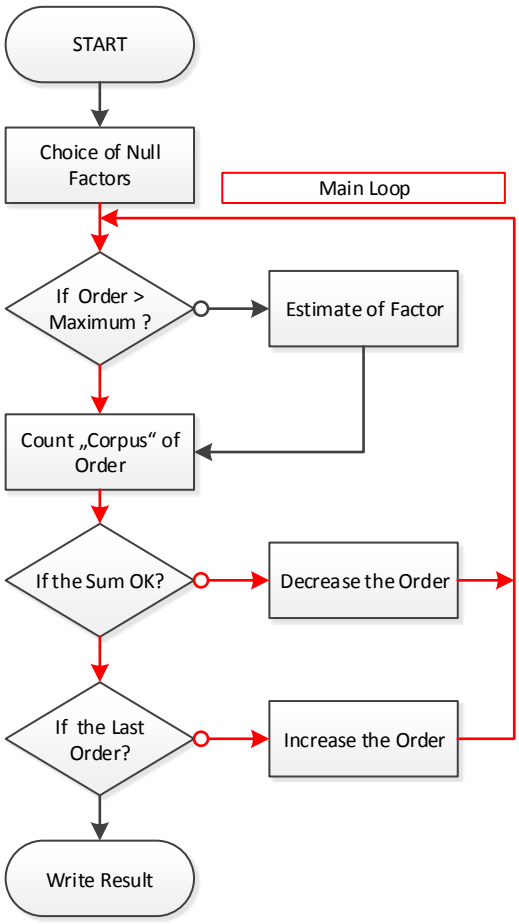

Figure 1: A Flowchart of program for factoring

Red arrows mark the working cycle of the program. Basically, in each step the program compares the final result with the assumption. If they correspond to one another, the program starts to calculate in higher order of magnitude. If the result does not correspond with the assumption, the program starts to calculate in lower order of magnitude where the combination of other factors is generated. Afterwards, the calculation proceeds to the higher order of magnitude again. If the attained order of magnitude is higher than the the reach of the order of magnitude higher than the maximum length of factors there would be no more factors estimated, only transmissions from lower orders of magnitude would be completed. If the program tests all the possibilities and no result is attained, the processed number is prime number.

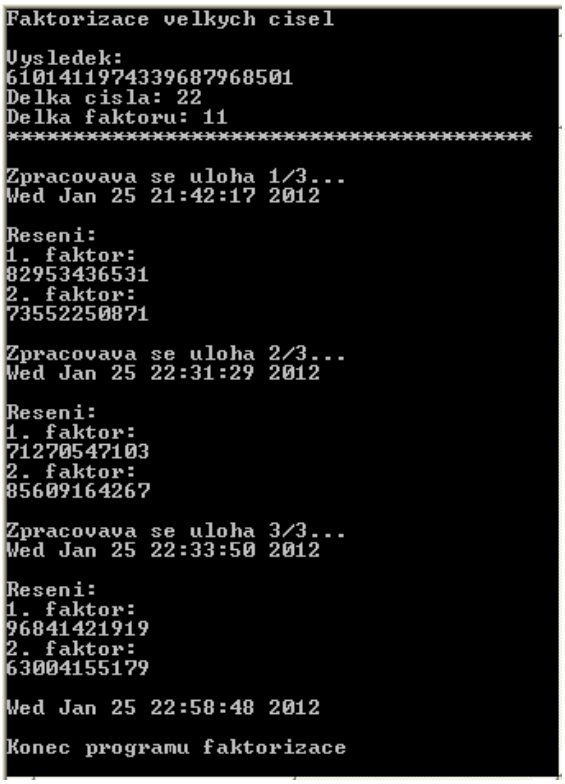

Figure 2: Demonstration of resulting console factorization 22-digit number

5. CONCLUSION

This article describes the formation of algorithm for factorization of large numbers. Figure 2 shows time needed for calculation of factorization 22-digit number (the time is 3000 sec). Obviously, this is not reasonable time for the practical use of this program. Program was made in language C++ in the environment of Microsoft Visual Studio 2010. When the program was on the run, the processor of the computer was fully occupied too, therefore it was impossible to perform any other tasks. Program was made for demonstrative and educational purposes and it definitely does not aim to be the significant tool for the break of cryptosystem RSA.

ACKNOWLEDGEMENT

The work was supported by the Brno University of Technology under project no. FEKT-S-14-2300: “New types of electronic circuits and sensors for specific applications”.

REFERENCES

[1] BURDA, K. Aplikovaná kryptografie (přednáška 7) Asymetrické kryptosystémy. VUT v Brně, rok neuveden

[2] Wikipedia: RSA. [online]. [cit. 2011-12-18]. Dostupné z: http://en.wikipedia.org/wiki/RSA

[3] RSA: RSA History. *RSA* [online]. [cit. 2011-12-18]. Dostupné z: http://www.rsa.com/node.aspx?id=2760