16. Twisted Edwards Curves / Bernstein D. J., Birkner P., Joye M., Lange T., Peters C. // Lecture Notes in Computer Science. 2008. P. 389–405. doi: https://doi.org/10.1007/978-3-540-68164-9_26
17. Bessalov A. V., Kovalchuk L. V. Exact Number of Elliptic Curves in the Canonical Form, Which are Isomorphic to Edwards Curves Over Prime Field // Cybernetics and Systems Analysis. 2015. Vol. 51, Issue 2. P. 165–172. doi: https://doi.org/10.1007/s10559-015-9709-x
18. Bessalov A. V., Dihtenko A. A. Cryptographically resistant Edwards curves over prime fields // Applied Radio Electronics. 2013. Vol. 12, Issue 2. P. 285–291.
19. Bespalov O. Yu., Kuchynska N. V. Kryva Edvardsa nad kiltsem lyshkiv yak dekartiv dobutok kryvykh Edvardsa nad skinchenymy poliamy // Prikladnaya radioelektronika. 2017. Vol. 16, Issue 3-4. P. 170–175.

Метод Ферма вважається кращим при факторизації чисел $N=p\cdot q$ у випадку близьких p і q. Обчислювальна складність базового алгоритму методу визначається кількістю пробних значень X при вирішенні рівняння $Y^2=X^2-N$, а також складністю арифметичних операцій. Для її зниження запропоновано в якості допустимих розглядати ті з пробних X, для яких $(X^2-N)\,\mathrm{mod}\,bb$ є квадратним залишком по модулю bb, названого базовым. При використанні базової основи модуля bb число пробних X зменшується в число раз, близьке до $Z(N,bb)=bb/bb^*$, де bb^* – число елементів множини T коренів рівняння $(Y\,\mathrm{mod}\,b)^2\,\mathrm{mod}\,b=((X\,\mathrm{mod}\,b)^2-N\,\mathrm{mod}\,b)\,\mathrm{mod}\,b$, а Z – коефіцієнт прискорення.

Визначено, що на величину Z(N,bb) впливають значення залишків $N\,\mathrm{mod}\,p$ (при p=2 використовуються залишки $N\,\mathrm{mod}\,8$). Запропоновано постановку задачі пошуку bb з максимальным Z(N,bb) при обмеженях на обсяг пам'яті ЕОМ, де визначаються показники степенів простих чисел – множників bb, та спосіб її вирішення.

Для зменшення числа арифметичних операцій з великими числами попонується замість таких виконувати операції зі значеннями різниць між найближчими значеннями елементів множини T. Тоді арифметичні операції множення і додавання з великими числами виконуються рідко. А якщо квадратний корінь з X^2-N визначати тільки у випадках, коли значення

$(X^2-N)\,\mathrm{mod}\,b$ будуть квадратними залишками для багатьох різних основ модуля b, то обчислювальною складністю цієї операції можна знехтувати.

Встановлено, що тоді запропонований модифікований алгоритм методу Ферма для чисел 2^{1024} забезпечує зниження обчислювальної складності в порівнянні з базовим алгоритмом в середньому в 10^7 раз

Ключові слова: факторизація, метод Ферма, обчислювальна складність, базова основа, проріджування, квадратні залишки

UDC 511:003.26.09
DOI: 10.15587/1729-4061.2018.150870

APPLICATION OF THE BASIC MODULE'S FOUNDATION FOR FACTORIZATION OF BIG NUMBERS BY THE FERMAT METHOD

S. Vynnychuk
Doctor of Technical Sciences,
Senior Researcher, Head of Department
Department of modeling of
energy processes and systems
Pukhov Institute for Modelling in Energy
Engineering National Academy of
Sciences of Ukraine
Henerala Naumova str., 15, Kyiv, Ukraine, 03164
E-mail: vynnychuk@i.ua
Y. Maksymenko
PhD*
E-mail: maksimenco@gmail.com
V. Romanenko
PhD, Head of Department*
E-mail: roma_38@ukr.net
*Institute of Special Communication
and Information Security
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
Verhnyoklyuchova str., 4, Kyiv, Ukraine, 03056

1. Introduction

At present, the issue of information security is one of the most relevant. One of the ways to solve it is information encryption. Among the ways of encryption, the asymmetric crypto-algorithm (ACA) RSA has acquired widespread application. Its cryptographic resistance is caused by the complexity of factorization of big numbers $N=p\cdot q$, where p and q are prime numbers. In papers [1, 2], it was shown that the known examples of compromising the RSA algorithm work only for its specific implementations, and, as a rule, in the general case are not most effective for solving a factorization problem.

Up to now, many factorization methods have been developed. The most frequently used methods include the methods of the number field sieve (GNFS), the quadratic sieve method (QS), the Pollard method and the Fermat method [3–6]. In this case, it is believed that each of these methods is the best (most effective in terms of computational complexity) for its application area. Thus, the Fermat method is most effective at sufficiently close values of prime factors p and q, The Pollard

ρ-method is most effective at a small enough value of one of the multipliers. Except for the regions of values of N, where the most effective methods are the Fermat method or the Pollard ρ-method, the method of quadratic sieve is most effective for $N<10^{110}$ [4], the method of the number field sieve is most effective at $N>10^{110}$. That is why the development of modifications of these methods that make it possible to reduce the computational complexity of any of these methods at the stage of cryptographic analysis of ACA RSA will ensure greater reliability of an encryption key. It should also be noted that both the QS and the GNFS are the variations of the factor foundation method, which is a generalization of the Fermat factorization method. That is why the Fermat method holds a special position among the well-known factorization methods, and the research, associated with decreasing computational complexity of the algorithm of its implementation can be relevant for other methods.

2. Literature review and problem statement

It is commonly known that the Fermat method is used only for factors p and q of number N that are close by values. The region of its application is quite narrow. The main ideas, associated with reduced computational complexity of the algorithm that implements it, were proposed and studied relatively long ago and are presented in paper [9].

According to the classic variant of the algorithm of the Fermat method [10, 11], to derive the values of p and q, the equation is solved

$$X^2 - N = Y^2, \tag{1}$$

where X and Y are positive integers.

Unknown X is represented in the form of

$$X_i = \left(\left[\sqrt{N}\right]+1\right) + x_i = x_0 + k. \tag{2}$$

The solution to equation (1) is obtained by searching the values of k=0, 1, 2,..., until residue $X^2 - N$ is complete square of integer. If solution (1) is obtained at

$$X^* = x_0 + x^*,$$

where

$$Y^* = \sqrt{\left(X^*\right)^2 - N},$$

p and q are determined according to ratios:

$$\begin{cases} p = X - Y, \\ q = X + Y. \end{cases} \tag{3}$$

The main disadvantage of the Fermat factorization method is the need for multiple performance of arithmetically complex operations of raising to square, subtraction and calculating the square root for big numbers, which determines its computational complexity. In this case, it is necessary to distinguish between the following components of the problem of high computational complexity of the basic algorithm:

1) a large number of X, for which the ratio (1) should be checked;

2) significant computational complexity of the operation of deriving square root of multidigit numbers;

3) high computational complexity of the operations of multiplication and addition of multidigit numbers.

In most of the known variants of reduction of computational complexity of the Fermat factorization method, the procedure of preliminary sifting either of analyzed values X, or reduction of the check operations of square root calculation is used. One of the ways of solving the first problem is based on the results of analysis of values m of lower bits of a factorized number [12]. Paper [9] considers the possibility of increasing the pitch of thinning, but the value of such a pitch is a permanent magnitude. Such permanent pitch can be equal to values of 2, 4, 6 and, in rare cases, of 12. However, it cannot significantly affect the reduction of computational complexity of the algorithm of the Fermat method. That is why the search for the ways of reduction of the number of check X, for which ration (1) is checked, is one of the tasks that are explored in this study.

The options of the solution of the second problem were proposed in [13, 14], where a reduction in the number of operations of square root calculation is achieved by the results of analysis of least significant bits y^2. A modified version of these algorithms is presented in [15]. In paper [16], the method for determining that the square root is not an integer without the procedure of root calculation was proposed.

In papers [17, 18], reduction of computational complexity of the Fermat factorization method is ensured through the use of modular arithmetic and the apparatus of continuous fractions, respectively.

In [9], it is proposed to check whether the difference X^2-N by module of a certain set of foundations of modules b that are prime numbers will be quadratic residue.

At satisfaction of ratio (1) for an arbitrary foundation of module, the equality will be satisfied

$$Y^2 \bmod b = \left(X^2 - N\right) \bmod b, \tag{4}$$

which is equivalent to satisfaction of the ratio

$$\left(Y \bmod b\right)^2 \bmod b = \\ = \left(\left(X \bmod b\right)^2 \bmod b - N \bmod b\right) \bmod b. \tag{5}$$

It should be noted that if the ratio (1) is satisfied, there is equality (4) and (5) for an arbitrary b. The opposite is false, that is, satisfaction of (1) does not follow from satisfaction of (5). However, if the ratio (5) is not satisfied, the ratio (1) will not be satisfied. That is why for the X, for which (5) is not satisfied, it may not be possible to derive square root, as it may not be the exact square of an integer.

Through the implementation of checks (5) for set $MB = \{b_k\}_{k=1}^{m}$ of foundations of modules, computational complexity of the Fermat method decreases. If each of the modules is a prime number, then, as noted in [9], when using one additional module in (5), the number of X, for which difference X^2-N can be complete square, decreases actually by two times. Such X will be subsequently called admissible.

However, when using the first foundation of modules, which will be subsequently called basic and designated bb, the number of X, for which the ratios (5) will be analyzed at other values of modules, will decrease only approximately by half.

Let us assume that bb is a foundation of module and bb^* is the number of roots of the equation (5) at $b=bb$. If subsequently during analysis of the check X, only bb^* of them will be analyzed, the number of analyzed X will decrease by the number of times equal to

$$Z(N, bb) = bb / bb^*, \quad (6)$$

where $Z(N, bb)$ will subsequently be called acceleration coefficient.

In the scientific literature, there are no methods of the use as the basic foundation of module *bb* of the numbers, which are products of primes numbers or powers of such prime numbers, at using of which the pitch for X will be non-uniform, and estimates of the value of *Z(N, bb)*. Such idea was proposed by the authors in papers [19–21], where the values of *bb* were determined from the condition of ensuring the lowest possible reduction in the number of admissible *X*, and the impact of the number of N on the number of admissible *X* in (5) at *b=bb* was not assessed. Such research is one of the problems that are being solved.

The hardware capabilities of calculations, such as graphic charts have considerably increased lately. While the algorithm of the Fermat method is easy to de-parallel, the task of number factorization could be performed using graphic processors. However, the types of data that are currently used in them do not imply the possibility of working with multidigit numbers. The methods for performing operations with numbers of *long* type instead of similar operations with big numbers are one of the most important tasks that are important when designing modifications of the Fermat algorithm.

That is why it is advisable to conduct studies regarding the use of the basic foundation of the module in ratios (5) both in terms of achieving a significant reduction in the number of admissible X, and for reducing computational complexity of the operations of multiplication and addition of multidigit numbers based on the operations with numbers of the *long* type.

3. The aim and objectives of the study

The aim of this study is to ensure the reduction of computational complexity of the algorithm of the Fermat method of factorization of big numbers using the basic foundation of the module that is the product of powers of prime numbers. This will make it possible to design hardware-software means of conducting cryptanalysis ACA that are more effective in terms of performance speed and, consequently, to enhance the quality of evaluation of crypto-resistance of ACA of RSA.

To accomplish the aim, the following tasks have been set:

– to establish what prime numbers (multipliers *bb*) influence the value of acceleration coefficient $Z(N, bb)$ at a fixed *N*, determined according to (6);

– to find out how the values of *Z(N, bb)* are influenced by numbers *N*;

– to offer the method to reduce the number of operations of multiplication and addition of multidigit numbers based on using the operation with the numbers of the *long* type when performing arithmetic operations with big numbers.

4. Analysis of influence of number N and exponents of prime numbers in the structure of basic foundation

The results of research into the impact of number N and powers of prime numbers in the structure *bb* were derived based on conducting numerical experiments. Analysis and generalization of the results are given below.

A general idea of the change in acceleration coefficient at changes of *bb* and a fixed value of *N* can be obtained based on the data of Tables 1, 2, which show the information for all $N \bmod bb < 60$, coprime with 2, 3 and 5.

Table 1

Value of acceleration coefficients $Z(bb, N \bmod bb)$ for various *bb* as products of number 60 on powers of 2, 3 and 5 for values of $N \bmod bb < 60$ coprime with 2, 3 and 5

$N \bmod bb$	Variants of values of *bb*											
	60	240	180	300	720	900	1,200	960	540	1500	8,640	24,000
	*1	*4	*3	*5	*12	*15	*20	*16	*9	*25	*144	*400
1	5	10	15	10.71	30	32.14	21.43	20	22.5	12.1	90	48.39
7	7.5	15	22.5	7.5	45	22.50	15.00	15	33.75	7.5	67.5	15.00
11	10	20	10	21.43	20	21.43	42.86	20	10	24.19	20	48.39
13	7.5	15	22.5	7.5	45	22.5	15	30	33.75	7.5	135	30
17	15	30	15	15	30	15	30	60	15	15	60	60
19	5	10	15	10.71	30	32.14	21.43	10	22.5	12.1	45	24.19
23	15	30	15	15	30	15	30	30	15	15	30	30
29	10	20	10	21.43	20	21.43	42.86	40	10	24.19	40	96.77
31	5	10	15	10.71	30	32.14	21.43	10	22.5	12.1	45	24.19
37	7.5	15	22.5	7.5	45	22.5	15	30	33.75	7.5	135	30
41	10	20	10	21.43	20	21.43	42.86	40	10	24.19	40	96.77
43	7.5	15	22.5	7.5	45	22.5	15	15	33.75	7.5	67.5	15
47	15	30	15	15	30	15	30	30	15	15	30	30
49	5	10	15	10.71	30	32.14	21.43	20	22.5	12.1	90	48.39
53	15	30	15	15	30	15	30	60	15	15	60	60
59	10	20	10	21.43	20	21.43	42.86	20	10	24.19	20	48.39

Table 2

Changes in acceleration coefficients $Z(bb, N \bmod bb)$ at changes in *bb* compared to *bb*=60 for $N \bmod bb < 60$, coprime with 2, 3 and 5

$N \bmod bb$	Variants of values of *bb*											
	60	240	180	300	720	900	1,200	960	540	1,500	8,640	24,000
	*1	*4	*3	*5	*12	*15	*20	*16	*9	*25	*144	*400
1	1	2	3	2.14	6	6.43	4.29	4	4.5	2.42	18	9.68
7	1	2	3	1	6	3	2	2	4.5	1	9	2
11	1	2	1	2.14	2	2.14	4.29	2	1	2.42	2	4.84
13	1	2	3	1	6	3	2	4	4.5	1	18	4
17	1	2	1	1	2	1	2	4	1	1	4	4
19	1	2	3	2.14	6	6.43	4.29	2	4.5	2.42	9	4.84
23	1	2	1	1	2	1	2	2	1	1	2	2
29	1	2	1	2.14	2	2.14	4.29	4	1	2.42	4	9.68
31	1	2	3	2.14	6	6.43	4.29	2	4.5	2.42	9	4.84
37	1	2	3	1	6	3	2	4	4.5	1	18	4
41	1	2	1	2.14	2	2.14	4.29	4	1	2.42	4	9.68
43	1	2	3	1	6	3	2	2	4.5	1	9	2
47	1	2	1	1	2	1	2	2	1	1	2	2
49	1	2	3	2.14	6	6.43	4.29	4	4.5	2.42	18	9.68
53	1	2	1	1	2	1	2	4	1	1	4	4
59	1	2	1	2.14	2	2.14	4.29	2	1	2.42	2	4.84

Based on an analysis of data from Tables 1, 2, it is possible to draw two major conclusions:

– at a change in *bb*, acceleration coefficient varies depending on the value of an additional multiplier in it;

–acceleration coefficients take a series of the same values for a set of magnitude of *N*mod*bb*, which at various *bb* is different.

Thus, at the increase in bb by 3 times (*bb*=180), acceleration coefficient increases either by three times, or remains unchanged. At an increase in *bb* by 9 times (*bb*=540), acceleration coefficient increases either by 4.5 times, or remains unchanged. In this case, an increase takes place for the same Nmod3 as at *bb*=180. If *bb* increases by 5 or by 25 times, acceleration coefficient also increases for the same *N*mod5. The increase in bb by $2^{c1}*3^{c2}*5^{c3}$ times leads to an increase in acceleration coefficient that is equal to the product of accelerations, related to an increase in bb of the exponent of number 2 by c1, of the exponent of number 3 by c2 and the exponent of number 5 by c3.

Thus, we can assume that for multipliers *bb* equal to p^t, it is possible to determine a set of values *N*mod*p*, for which the values of acceleration coefficients do not change at the change of exponent, and those for which they change. This assumption was checked using numerical experiments for multipliers *bb* – prime p from *p*=2 to *p*=31. The results of such research are given below.

a) Multiplier *bb* *p*=2. Based on numerical experiments, it was found that for 2^t, it is advisable to use the value of exponent $t\geq 2$, since at *t*=1, *N*mod2 of acceleration is equal to 1. Table 3 shows the values of acceleration coefficients for all odd values of *N*mod2^t at *t*=3÷7 that are coprime with *bb*.

Based on data from Table 3, it is possible to make a conclusion that the character of a change in the values of acceleration coefficients for $bb=2^t$ at $t>3$ is determined by magnitude of residue *N*mod8, which was proved by additional numerical experiments with $bb=2^t$ at $t\leq 14$. For such values of *bb* at *t*=1÷14, Table 4 shows the values of acceleration coefficients depending on Nmod8.

Table 4

Acceleration coefficients *Z*(*bb*, *N*mod*bb*) for odd *N*mod8 at $bb=2^t$ *t*=1÷14

*N*mod8	1	3	5	7	*N*mod8	1	3	5	7
t=1	1	–	–	–	*t*=8	16	4	8	4
t=2	2	2	–	–	*t*=9	18.2857	4	8	4
t=3	2	4	2	4	*t*=10	21.3333	4	8	4
t=4	4	4	4	4	*t*=11	22.2609	4	8	4
t=5	4	4	8	4	*t*=12	23.2727	4	8	4
t=6	8	4	8	4	*t*=13	23.5402	4	8	4
t=7	10.6667	4	8	4	*t*=14	23.8140	4	8	4

According to data from Table 4, the magnitude of acceleration coefficient for $bb=2^t$ at $t>2$ is determined by exponent *t* and the value of *N*mod8. This is proved by the data in Table 3 at *t*=3÷7, where, for example, at *t*=7 and *N*mod8=1 acceleration coefficients for values of *N*mod2^7, equal to 1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105 and 113 will be the same (equal to 32/3), that is, for those that (*N*mod2^7)mod8=1. That is why the data from Table 4 allow estimating the constructions of the effective primary base *bb* for the cases when there is number 2 among prime multipliers *bb*. Thus, at *N*mod8=3 and *N*mod8=7 at $bb=2^t$ and $t\geq 3$, acceleration coefficient always equals to 4 and there is no point using power 2 with the exponent higher than 3 in bb. If *N*mod8=5, it will be optimal to use power 2 with exponent 5 in bb. But if *N*mod8=1, it is possible to use power 2 with exponent 8 and more in *bb*.

b) Multiplier bb *p*=3. In case multiplier 3 is included in *bb*, it was found that at *N*mod3=2, the values of acceleration coefficients for $bb=3^t$ at $t\geq 1$ coincide, which is proved by numerical experiments with $bb=3^t$ at *t*=1÷8. For such values of bb, Table 5 gives values for acceleration coefficients, depending on Nmod3 at *t*=1÷8.

Table 3

Acceleration coefficients *Z*(*bb*, *N*mod*bb*) for odd *N*mod2^t at *t*=3÷7

bb	Acceleration *Z*=*Z*(*bb*, *N*mod*bb*) for all possible values of *N*mod*bb*																
8	*N*mod*bb*	1	3	5	7	–	–	–	–	–	–	–	–	–	–	–	–
	Z	2	4	2	4	–	–	–	–	–	–	–	–	–	–	–	–
16	*N*mod*bb*	1	3	5	7	9	11	13	15	–	–	–	–	–	–	–	–
	Z	4	4	4	4	4	4	4	4	–	–	–	–	–	–	–	–
32	*N*mod*bb*	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
	Z	4	4	8	4	4	4	8	4	4	4	8	4	4	4	8	4
64	*N*mod*bb*	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
	Z	8	4	8	4	8	4	8	4	8	4	8	4	8	4	8	4
	*N*mod*bb*	33	35	37	39	41	43	45	47	49	51	53	55	57	59	61	63
	Z	8	4	8	4	8	4	8	4	8	4	8	4	8	4	8	4
128	*N*mod*bb*	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
	Z	10.7	4	8	4	10.7	4	8	4	10.7	4	8	4	10.7	4	8	4
	*N*mod*bb*	33	35	37	39	41	43	45	47	49	51	53	55	57	59	61	63
	Z	10.7	4	8	4	10.7	4	8	4	10.7	4	8	4	10.7	4	8	4
	*N*mod*bb*	65	67	69	71	9	75	77	79	81	83	85	87	89	91	93	95
	Z	10.7	4	8	4	10.7	4	8	4	10.7	4	8	4	10.7	4	8	4
	*N*mod*bb*	97	99	101	103	105	107	109	111	113	115	117	119	121	123	125	127
	Z	10.7	4	8	4	10.7	4	8	4	10.7	4	8	4	10.7	4	8	4

According to Table 5, it is possible to estimate the possibilities of construction of the effective primary base of *bb* for cases when there is number 3 among prime multipliers. Thus, at *N*mod3=2 at $bb=3^t$ and $t>0$, acceleration coefficient always equals to 3. There is no point using power 3 with the exponent higher than 1 in *bb*. If *N*mod3=1, it is possible to use power 3 with exponent 4 and more in *bb*.

Table 5

$Z(3^t, N\text{mod}3)$ for coprime with 3 *N*mod3 at $bb=3^t$ and $t=1\div8$

*N*mod3	1	2	*N*mod3	1	2
$t=1$	1.5	3	$t=5$	11.0455	3
$t=2$	4.5	3	$t=6$	11.7581	3
$t=3$	6.75	3	$t=7$	11.8859	3
$t=4$	10.125	3	$t=8$	11.9726	3

c) Multiplier *bb* $p=5$. If multiplier 5 is included in *bb*, it was found that the values of acceleration coefficients for $bb=5^t$ at $t\geq1$ coincide for all Nmod5=2 and *N*mod5=3, which was proved by the numerical experiments with $bb=5^t$ at $t=1\div6$. But at $t>1$ and *N*mod5=1 and *N*mod5=4, the value of acceleration coefficient increases. Such value at every t is the same at *N*mod5=1 and *N*mod5=4. For such values of *bb,* Table 6 shows the values of acceleration coefficients, depending on *N*mod5.

Table 6

$Z(5^t, N\text{mod}5)$ for *N*mod5 coprime with 5 at $bb=5^t$ and $t=1\div6$

*N*mod5	1	2	3	4	*N*mod5	1	2	3	4
$t=1$	1.6667	2.5	2.5	1.6667	$t=4$	4.2517	2.5	2.5	4.2517
$t=2$	3.5714	2.5	2.5	3.5714	$t=5$	4.2750	2.5	2.5	4.2750
$t=3$	4.0323	2.5	2.5	4.0323	$t=6$	4.2843	2.5	2.5	4.2843

Based on data from Table 6, it is possible to assess more accurately the possibilities of the effective primary base of bb for the cases when there is number 5 among prime multipliers Thus, at *N*mod5=2 or *N*mod5=3 at $bb=5^t$ and $t>0$

$$Z(5^t, N \bmod 5) = 2.5.$$

That is why it is advisable to use exponent $t=1$. If *N*mod5=1 and *N*mod5=4, it is possible to use in *bb* power 5 with exponent 2 and more.

d) Multiplier *bb* $p=7$. If multiplier 7 is included in *bb*, it was found that the values of acceleration coefficients for $bb=7^t$ at t 1 coincide for all *N*mod7=3, *N*mod7=5 and *N*mod7=6, which was proved by numerical experiments with $bb=7^t$ at $t=1\div3$. But at $t>1$ and *N*mod7=1 *N*mod7=2 and *N*mod7=4, the value of acceleration coefficient increases and takes the same value. For such values of *bb,* Table 7 shows the acceleration coefficients depending on *N*mod7 at $t=1\div4$.

Based on data from Table 7, it is possible to assess the possibilities of constructing the effective primary base of *bb* for the cases when there is number 7 among prime multipliers of *bb*. Thus, at *N*mod7=3, *N*mod7=5 or *N*mod7=6 at $bb=7^t$ and $t>0$, acceleration coefficient is always equal to 2.3333. That is why there is no point using in bb power 7 with exponent more than 1. If *N*mod7=1, *N*mod7=2 or *N*mod7=4, it is possible to use in *bb* power 7 with exponent 2 and more.

Table 7

Acceleration coefficients for *N*mod7, coprime with 7

*N*mod7	1	2	3	4	5	6
$t=1$	1.75	1.75	2.3333	1.75	2.3333	2.3333
$t=2$	3.0625	3.0625	2.3333	3.0625	2.3333	2.3333
$t=3$	3.2358	3.2890	2.3333	3.2890	2.3333	2.3333

e) Multipliers *bb* $p>7$ and $p\leq23$. For prime p – multipliers of *bb*, equal to 11, 13, 17, 19, 23, it was found that the character of a change in the values of acceleration coefficients for $bb=p^t$ at $t\geq1$ is determined by the value of Nmodp, which was proved by numerical experiments with $bb=p^t$ at $t=1\div4$. For such values of *bb,* Table 8 shows the values of acceleration coefficients, depending on *N*mod$p$ at $t=1, 2$.

Table 8

Values $i=N\text{mod}p$, for which $Z(p, 1)=Z(p^2, i)$ and $Z(p, i)<Z(p^2, i)$

p	Values $i=N\text{mod}p$, for which $Z(p, i)=Z(p^2, i)$	t	$Z(p^t, i)$	Values $i=N\text{mod}p$, for which $Z(p, i)<Z(p^2, i)$	t	$Z(p^t, i)$
11	2, 6, 7, 8, 10	1	2.2000	1, 3, 4, 5, 9	1	1.8333
		2	2.2000		2	2.6300
13	2, 5, 6, 7, 8, 11	1	2.1667	1, 3, 4, 9, 10, 12	1	1.8571
		2	2.1667		2	2.5224
17	3, 5, 6, 7, 10, 11, 12, 14	1	2.1250	1, 2, 4, 8, 9, 13, 15, 16	1	1.9000
		2	2.1250		2	2.3884
19	2, 3. 8, 10, 12, 13, 14, 15, 18	1	2.1111	1, 4, 5, 6, 7, 9, 11, 16, 17	1	1.8889
		2	2.1111		2	2.3442
23	5, 7, 10, 11, 14, 15, 17, 19, 20, 21, 22	1	2.0909	1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18	1	1.9167
		2	2.0909		2	2.2902

We will show on the examples of numbers *N*, presented in Table 9, that taking into consideration the specifics of numbers *N*, namely, the residues of dividing *N* by 8 and by prime p from 3 to 23, makes it possible to construct a new *bb*, at which set *D(bb, N)* will contain the number of admissible *X,* which does not exceed its value for bb=277200, equal to 2880 cells of memory of type *int*. The primary foundation for bb=277200 is the product of powers of prime numbers 2, 3, 5, 7, 11: $bb=2^4*3^2*5^2*7*11$ and is characterized by the value of acceleration coefficient for numbers *N* of magnitude of 96.25, shown in Table 9. According to the data given in Tables 4–8, we will construct new, more effective *bb′*. The data are shown in Table 9.

Table 9

Values of $N_j \bmod p$ for $p=2^3=8$ and prime $p=3\div23$

j	*N*	*p*								
		8	3	5	7	11	13	17	19	23
1	2 190 107 742 436 404 740 487 152 427 983	7	2	3	2	5	1	5	15	22
2	115 103 357 258 699 743 681 239 319 283	3	2	3	1	5	1	13	11	17
3	24 197 500 008 691 435 623 032 029 847	7	2	2	2	4	3	13	16	12
4	7 193 959 711 947 061 718 333 522 687	7	2	2	1	9	2	1	10	2
5	3 024 687 551 113 421 119 054 532 273	1	2	3	2	1	12	4	13	21
6	1 798 489 957 219 681 011 882 800 933	5	2	3	1	3	1	13	13	1

Taking into consideration the above recommendations on the selection of exponents of prime p – multipliers *bb*, Table 10 gives the refined values of *bb* , which take into account the specificity of factorized number *N*.

Table 10

Refined primary foundations of bb', formed for numbers N with respect to data from Tables 4–6

j	Exponents for prime numbers that make up new primary foundation									bb'	z	Memory amount bb'/z
	2	3	5	7	11	13	17	19	23			
1	3	1	1	2	1	1	–	–	–	840,840	312.812	2,688
2	3	1	1	2	1	1	–	–	–	840,840	312.812	2,688
3	3	1	1	2	1	1	–	–	–	840,840	312.812	2,688
4	3	1	1	2	1	1	–	–	–	840,840	364.948	2,304
5	10	1	1	2	–	–	–	–	–	752,640	490	1,536
6	5	1	1	2	–	–	–	1	–	446,880	387.917	1,152

As it follows from data in Table 10 for refined bb', due to taking into account the specificity of number N, the amount of required memory of the computer decreased and the value of the acceleration coefficient simultaneously increased by 3.25÷5.091 times, which approximately reduces factorization time by the same number of times. That is why it is advisable to consider the problem of searching for the optimal bb that takes into consideration the specificity of numbers N.

5. Determining the optimal primary foundation of bb taking into account the specificity of the factorized number

When setting the problem of searching for the optimal primary base of module bb, we will use the information about the structure of bb, about the properties of acceleration coefficients and the number of elements of set $D(bb, N\mathrm{mod}bb)$:

$$bb=\prod_{i=1}^{h} p_i^{k_i}; \tag{9}$$

$$Z(bb,N)=\prod_{i=1}^{h} z\left(p_i^{k_i},N\right); \tag{10}$$

the number of elements of array $D(bb, N\mathrm{mod}bb)$ is equal to:

$$bb / Z(bb,N)=\prod_{i=1}^{h} p_i^{k_i} / Z(p_i^{k_i},N). \tag{11}$$

According to (9) to (11), to determine bb, it is sufficient to determine the exponents of prime numbers – multipliers bb, where it is necessary to consider the relationship between the value of a prime number and an increase in acceleration at an increase in the exponent. For prime p from 2 to 23, the corresponding values of acceleration coefficients are determined according to data from Tables 4–8. When setting the task of searching for optimal bb with consideration of N, we will explore the possible types of the variants of values of exponents, depending on p, among which there will be an option when multiplier p is not used in bb and then $z(p^0, N)=1$.

For $p=2$, three types of variants are possible:

1) at $N\mathrm{mod}8=3$ or $N\mathrm{mod}8=7$, exponent t is always equal to 3;

2) at $N\mathrm{mod}8=5$, exponent t is always equal to 5;

3) at $N\mathrm{mod}8=1$, it is necessary to determine the value of exponent t.

In case if prime $p>2$, it is also necessary to consider three types of variants:

1) Nmodp takes such value that $Z(p, N\mathrm{mod}p)=Z(p^2, N\mathrm{mod}p)$ and $t=1$;

2) $t=0$ and $Z(p^0, N)=1$ (multiplier p is not used in bb);

3) $N\mathrm{mod}p$ takes such value that $Z(p, N\mathrm{mod}p)<Z(p^2, N\mathrm{mod}p)$ and $t\geq 1$.

Thus, when choosing the exponent of prime p – multiplier of bb, the exponent is not determined only for the third variant. To assess the possible range of exponents in variant 3, we will use the function of relative increase in acceleration coefficient, reduced to a memory unit:

$$s(p, t)=(Z(p^{t+1}, 1)/Z(p^t, 1)-1)/p, \tag{12}$$

which makes it possible to give an approximate estimate of the effectiveness of the primary base of module, related to additional multiplying bb by prime multiplier p. The values of function $s(p, t)$ for prime $p\geq 2$ and $p\leq 31$ for the series of variants of exponent are shown in Table 11.

Table 11

Values of function $s(p, t)$ for prime $p\geq 2$ i $p\leq 31$

p	t	$Z(p^t, 1)$	$Z(p^{t+1}, 1)$	$s(p, t)$	p	t	$Z(p^t, 1)$	$Z(p^{t+1}, 1)$	$s(p, t)$
2	5	4	8	0.5	7	0	1	1.75	0.10714
	6	8	10.66667	0.16667		1	1.75	3.0625	0.10714
	7	10.66667	16	0.25		2	3.0625	3.2358	0.00808
	8	16	18.2857	0.07143		3	3.2358	3.2890	0.002349
	9	18.2857	21.3333	0.08333	11	0	1	1.8333	0.07576
	10	21.3333	22.2609	0.02174		1	1.8333	2.6300	0.03953
	11	22.2609	23.2727	0.02273	13	0	1	1.8571	0.06593
	12	23.2727	23.5402	0.00575		1	1.8571	2.5224	0.02755
3	0	1	1.5	0.16667	17	0	1	1.8889	0.05229
	1	1.5	4.5	0.66667		1	1.8889	2.3884	0.01556
	2	4.5	6.75	0.16667	19	0	1	1.9000	0.04737
	3	6.75	10.125	0.16667		1	1.9000	2.3442	0.01230
	4	10.125	11.0455	0.03031	23	0	1	1.9167	0.03986
	5	11.0455	11.7581	0.02151		1	1.9167	2.2902	0.00847
5	0	1	1.6667	0.13333	29	0	1	1.9333	0.03218
	1	1.6667	3.5714	0.22857		1	1.9333	2.2190	0.00510
	2	3.5714	4.0323	0.02581	31	0	1	1.9375	0.03024
	3	4.0323	4.2517	0.01088		1	1.9375	2.2041	0.00444

Sorting values $s(p, t)$ in the descending order makes it possible to assess how effective the addition of multiplier p in bb will be. The more $s(p, t)$, the higher the effectiveness. If $s(p, t)$ is close to zero, at an increase in bb, acceleration coefficient increases slightly, but the amount of memory of a computer used to store increments for admissible X increases significantly. To search for the optimal bb taking into consideration the specificity of N and the methods for reducing computer memory, ratios (9) to (11) and limitations for the memory amount of the computer are used. The search for a maximum acceleration coefficient is through the search of admissible options of exponents of prime p – multipliers N.

In the numerical calculations on determining the optimal bb with consideration of the admissible memory amount needed to store increments of admissible X, it was accepted that the primary base of module bb is the product of powers

of prime numbers p equal to 2, 3, 5, 7, 11, 13, 17, 19, 23 29, 31, where the set of options of exponent of prime numbers p – multipliers bb is selected based on the data from Tables 5–8 and 11 on condition that $s(p, t)>0.03$ (t is an exponent for p). In this case, the following options of the influence on acceleration coefficient:

– for p=2 by Nmod8, we selected one of the possible types of variants, where in case if Nmod8=1, exponents of t=3÷12 were considered;

– for $p\geq 3$ and $p\leq 31$, two type of variants were selected: type 2 as well as one of the types 1 or 3, depending on the value of Nmod$p$.

In addition, when determining the required memory amount of the computer, it was taken into account that bb is always divided into 4, that is, the cyclic sequence of increments for bb is repeated at least twice.

In the numerical experiments for memory amount (magnitude $Q_{\max}$) 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, for each of the variants of the influence on acceleration coefficients, the maximum value of acceleration coefficient was determined. Since the number of such variants turns out to be quite large (equal to $3*2^8$=768), Table 12 shows the data only about $Z_{\min}$, $Z_{\max}$, and mean Z_{cp}, which is equal to the mean value for all the variants, where coefficient 2 is assigned for reaching the equality of the variants at p=2 and the type of variant 1, and coefficient 1 is given for variants 2 and 3. Then the weighted sum of all the obtained maximum acceleration coefficients was divided by 1,024. Derived values of $Z_{\min}$, $Z_{\max}$ and Z_{cp} were shown in Table 12 in the form of a diagram in Fig. 1.

Table 12

Values: $Z_{\min}$, $Z_{\max}$, Z_{cp} and necessary memory amount of computer $Zq(Z)$ for different boundary values $Q_{\max}$

$Q_{\max}$	$Z_{\min}$	$Zq(Z_{\min})$	$bb(Z_{\min})$	$Z_{\max}$	$Zq(Z_{\max})$	$bb(Z_{\max})$	Z_{cp}
10^2	89.610	77	13,800	472.5	96	90,720	214.520
10^3	213.571	924	394,680	1,386	960	2,661,120	579.731
10^4	405.786	9240	7,498,920	3,003	8,640	51,891,840	1,365.952
10^5	822.833	92,736	152,612,460	9,572.063	92,160	1,764,322,560	2,976.528
10^6	1,563.382	927,360	2,899,636,740	20,207.687	829,440	33,522,128,640	6,007.030
10^7	2,407.279	8,814,960	42,440,137,740	42,252.438	9,123,840	771,008,958,720	11,088.624

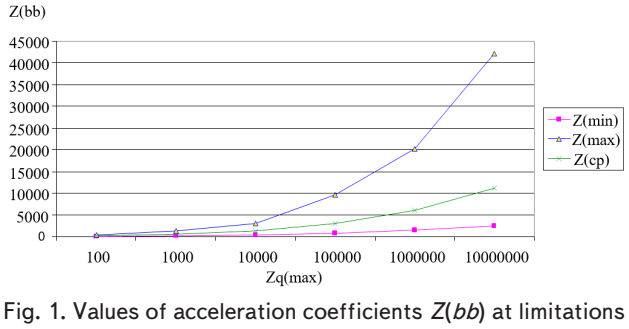

Fig. 1. Values of acceleration coefficients $Z(bb)$ at limitations for amount $Q_{(\max)}$ of available memory

According to data from Table 12, when using the optimal values of the basic foundation of the module, determined from the condition of the existing memory volume of computer, which does not exceed 10^7 cells of the *long* type, the number of check X will decrease in comparison with the basic algorithm of the Fermat method by 2.4×10^3÷ ÷4.2×10^4 times, where the mean value equals to 1.1×10^4 times. The computational complexity of the basic algorithm of the Fermat method will decrease by the same number of times. Further reduction of computational complexity can be achieved by reducing the number of operations of multiplication and subtraction of the numbers that exceed boundary values for the *long* type, which is considered below.

6. A decrease in the number of arithmetic operations with the numbers that exceed boundary values for the *long* type data

The algorithm of the Fermat method implies two basic operations for the check X: calculation of difference X^2-N and checking whether this difference will be the square of an integer. Based on using only the values X, which are admissible for bb, as check values, the number of check X decreased. However, this does not exclude performance of the operations of the calculation of difference X^2-N and checking whether it will be the square of an integer. In addition, the roots of the equation (5) at $b=bb$ can be big numbers, which can be seen in data from Table 12. That is why during the implementation of the modified algorithm of the Fermat method, it is proposed:

1. To use increments – differences of two nearest values of the roots of the equation (5) instead of the roots of the equation (5) at $b=bb$.

2. For each of the foundations of the modules of set $MB=\{b_k\}_{k=1}^m$ to determine the roots of equation (5) at $b=b_k$ (k=1÷m) and form array M_K of features for numbers from 0 to b_k–1, in which 1 will mean that $(X^2-N)\bmod b_k$ is the square residue, but zero, which is not true.

The first proposal makes it possible to represent the current admissible for bb check X in the form of:

$$X_{i+1}=X_i+\Delta x_i=$$
$$=X^*+\sum_{j=1}^{i}\Delta x_j=X^*+\Delta X_i, \qquad (13)$$

where X_{i+1} (X_i) is the following (previous) check X, admissible for bb; Δx_i is the increment for current admissible X; X^* is some intermediate fixed value, admissible for bb, which changes when magnitude ΔX_i is close to the boundary for the data of the *long* type. In such cases, we perform operations: $X^*=X^*+\Delta X_i$, $\Delta X_i=0$, as well as calculate residues $s_k=X^*\bmod b_k$ (k=1÷m).

Another proposal makes it possible to significantly decrease the number of operations of root calculation in relation to the basic algorithm of the Fermat method. To do this, in assessing the possibility that difference X^2-N can be complete square, the values are calculated

$$X\bmod b_k=(X^*+\Delta X_i)\bmod b_k=$$
$$=(s_k+\Delta X_i)\bmod b_k=r_{k,i}(k=1\div m), \qquad (14)$$

where the sum $s_k+\Delta X_i$ is the number of the *long* type, and numbers $r_{k,i}$ are smaller than b_k. It allows finding value $M_K[r_{k,i}]$. If $M_K[r_{k,i}]=0$, for the current check X the difference X^2-N cannot be a complete square and the transition to a new check X, admissible for bb, is performed. At $M_K[r_{k,i}]=1$, a similar magnitude for k+1-*th* module from set $MB=\{b_k\}_{k=1}^m$.

Under condition that value $M_K[r_{k,i}]=1$ for all $k=1\div m$, we calculate square root of X^2-N.

At this algorithm, the calculation of the value of check X (multidigit number) is performed in two cases:

– during calculation of the root of X^2-N, when X^2-N; is calculated;

– during recalculation of $X^*=X^*+\Delta X_i$, where ΔX_i is the number of the *long* type.

Since at a sufficient number of modules in set $MB=\{b_k\}_{k=1}^m$ the first variant occurs so rarely that it can be neglected, the computational complexity of the proposed modified algorithm is determined by the operations of assigning value $X^*+\Delta X_i$ to X^*, where ΔX_i is a *long* type number and by calculations of residues $s_k=X^* \bmod b_k$ ($k=1\div$m).

When presenting multidigit numbers by its coefficients for the base 1,000 or 1,024, during calculating the value of $X^*+\Delta X_i$, on average 4÷5 operations of addition the numbers of *long* type and 8÷10 operations of division of the sum by the foundation will be performed. We will estimate how often the cases of computation of values $X^*+\Delta X_i$ occur. The mean value of increments Δx_i is estimated by the magnitude of acceleration coefficient. If we use the optimal values of *bb*, obtained at restrictions for the amount of available memory of the computer of the order of 10^7 cells of the long type, it is possible to reach the magnitude of the boundary value of the number of long type ($\approx 2.147\times 10^9$) by the number of steps in the range from 50,000 to 900,000, where the average number of steps is equal to $2.147\times 10^9/11{,}088.624\approx 193{,}630$.

Now we will estimate the average number of operations with numbers of long type for values of N close to 2^{1024}, performed in the proposed modified Fermat method at the number of check values for its basic algorithm, equal to 2.147×10^9, when computation complexity of calculation operations X^2-N and the root from X^2-N can be neglected.

For the modified algorithm of the method, we have:

1. One operation $X^*+\Delta X_i$; on average 4–5 operations of addition of the numbers of the long type and 8–10 operations of dividing the sum by the foundation will be performed.

2. During performing operation $X^*+\Delta X_i$ one time, we calculate the values of

$$(X^*+\Delta X_i)\bmod b_k=(s_k+\Delta X_i)\bmod b_k \quad (k=1\div m),$$

where numbers s_k, ΔX_i and the sum $sk+\Delta X_i$ do not go beyond the boundaries of data of long type. That is the summary number of operations of m additions and m calculations of residues

$$(s_k+\Delta X_i)\bmod b_k (k=1\div m).$$

Primary values of $s_k(k=1\div m)$ are calculated before the entrance of check values X to the main cycle of the search of admissible values for *bb* and are not taken into account here.

3. On average, around 1.95×10^5 operations $\Delta X_i=\Delta X_{i-1}+\Delta x_i$, each of which is performed for *long* type numbers, which during representations of multidigit numbers by the array of coefficients, during factorization by foundation 1,024 requires every time on average 2 operations of addition and division.

The total average number of operations with *long* type numbers at m<100: addition – around 4×10^5, division and determining residue of division – around 4×10^5, which is approximately equal to $2\times 2.147\times 10^9/Z(N, bb)$.

1. 2.147×10^9 operations of increasing X by unity, complexity of which will be neglected.

2. 2.147×10^9 operations calculate the value of X^2-N. We will consider that due to the ratio

$$(X+1)^2-N=X^2-N+2X+1=Y+2X+1.$$

Computation complexity of determining the value of X^2-N can be the magnitude of order of $O(\log N)$.

3. 2.147×10^9 operations of calculation of square root, calculation complexity for which is estimated by magnitude $O(\log^2 N)$.

Thus, on average, for one check value of X among 2.147×10^9 operations, the modified algorithm requires $2/Z(N, bb)$ operations of adding and dividing *long* type numbers (including division with residue), and the basic algorithm of the Fermat method – $O(\log N+\log^2 N)$ operations. If we consider that the mean value of $Z(N, bb)$ is equal to 1.1×10^4, and N is close to 2^{1024}, computational complexity of the modified algorithm of the Fermat method decrease not less than by 10^7 times.

7. Discussion of results of reducing the computational complexity of the modified algorithm of the Fermat factorization method

An analysis of the factors that essentially influence computational complexity of the algorithm of the Fermat factorization method and its modifications revealed that they may include:

a) a relative number of values of k in the ratio (2), at which it is possible to establish beforehand that $(x_0+k)^2-N$ will not be square of an integer (mean value is (z(N,bb)–1)/z(N,bb));

b) a relative number of values of k in the ratio (2), at which based on checking the ratios (4) for a series of foundations of modules, it is possible to establish that $(x_0+k)^2-N$ will not be the square of integer (mean value is $(1/z(N, bb)*(1-2^{-m})$, where m is the number of integers in coprime modules that are used in the ratios (4));

c) a relative number of values of k in the ratio (2), at which if the ratios (4) for the selected set of values of foundations of modules are satisfied, it should be checked it multidigit number $(x_0+k)^2-N$ will be the square of the integer based on calculation of the root from it (mean value is equal to $(1/z(N, bb)*2^{-m}$, where m is the number of prime numbers in coprime modules, which are used in the ratios (4));

d) a relative number of steps of the algorithm, at which it is necessary to perform operations with multidigit numbers.

The studies, the results of which are presented in the article, were directed at achieving maximum values for the factor a) (that is, a maximum value of acceleration coefficient $z(N, bb)$), as well as the smallest possible number of steps, at which the operations with multidigit numbers (factor g) are performed.

To obtain maximum values of accelerated coefficient, we found the ratio (10) for $z(N, bb)$ and (11) for the amount of memory required to store the increments to admissible X as the difference between their two consecutive values (in ascending order). Based on them, we stated the problem of determining the optimal primary foundation of *bb* with consideration of specificity of the factorized number, in which one of the conditions is a restriction for the amount of avail-

able computer memory. The use of increments to admissible X instead of their values made it possible to replace most of the operations, which in the algorithm of the Fermat method are performed with multidigit numbers, with the operations with *long* type numbers. It is possible to solve the challenges, represented in section 2.

It should be noted that in the modified algorithms of the Fermat method, presented in [9], the problems concerning factor b) are solved.

The amount of available memory of a computer (or distributed computing systems) is the main constraint in practical use of the proposed modified Fermat algorithm, because the following problems are solved based on this information:

– determining the optimal primary foundation of bb taking into consideration the specificity of N, which corresponds to maximum value of $z(N, bb)$;

– determining a set of foundations of modules that are used to check the ratios (4), where the information about square residues is stored for each of the foundations.

The restrictions can also include the fact that at $z(N, bb)>1000$, its increase by two times is possible, as a rule, when a new integer appears in bb. And since bb is the product of powers of prime numbers, it is impossible in practice to obtain acceleration coefficients $Z_{max}>10^7$, let along $Z_{min}>10^7$ or $Z_{cp}>10^7$.

The presented solutions and the general algorithm of the modified Fermat algorithm were focused at the possibilities of their use both for single-processor computers, and for modern high performance distributed computation systems. In the latter case, it is important to take into consideration the amount of available memory for each of the processors, which was not explored in the study.

Methods of factorization of multidigit numbers can be considered as an iteration procedure, by which the satisfaction of a certain condition is checked at the check step k. In the case of the Fermat method, the condition is checked if the difference $\left(x_0+k\right)^2-N$ will be the square of an integer. In this case, it was found for the Fermat method, that it is not necessarily to determine the square root of (the conditions of factor c)), and in most cases it is enough to use the algorithms of thinning check values (factors a) and b)). It is possible to set the task of finding the ways to use this idea for other factorization methods as well. However, in the case of each of the factorization methods, it is necessary to identify conditions that should be checked and find the ways of thinning the check k.

8. Conclusions

1. It was established that powers of prime numbers – multipliers of bb form an integral of acceleration coefficient, which does not depend on other prime numbers or powers of such numbers, and total acceleration coefficient $Z(N, bb)$ at a fixed N is the product of such coefficients for powers of prime numbers.

2. Based on numerical experiments, it was found that depending on residues of dividing N by prime numbers – multipliers N, for each of the prime numbers p, sets $N\mathrm{mod}p$ are formed, for which at the fixed bb, acceleration coefficient $Z(N, bb)$ takes one and the same value. For prime numbers p from 3 to 31, it was shown based of numerical experiments that there are only 2 groups of values Nmodp. For the first of them, containing the values $N\mathrm{mod}p=1$, inequality $Z(N, p^2)>Z(N, p)$ is true, and for the second one, equality $Z(N, p^k)=Z(N, p)$, where $k>1$ is true. Three groups are formed for powers of prime $p=2$:

1) $N\mathrm{mod}8=1$;

2) $N\mathrm{mod}8=5$;

3) $N\mathrm{mod}8=3$ and $\mathrm{Nmod}8=7$.

Then, $Z(N,2^{5+k})>Z(N,2^5)$ for variant 1, $Z(N, 2^{5+k})=Z(N, 2^5)$ for variant 2, $Z(N, 2^{3+k})=Z(N, 2^3)$, where $k>0$ for variant 3. This made it possible to significantly reduce the number of possible variants of values of bb while solving the problem of determining its optimal value.

3. When using large values of basic foundation of module, it proved to be appropriate to represent the roots of the equation (5) through the difference between two consecutive values (in ascending order). This made it possible in most cases to perform operations with long-type numbers instead of multidigit numbers.

4. Based on the obtained results, we described the modified algorithm of the Fermat method, which in comparison with the basic algorithm ensures the reduction of computational complexity on average no less than by 10^7 times for the numbers of 2^{1024} order when using the optimal values of $Z(N, bb)$, provided that it is possible to record up to 10^7 of the long-type numbers in the computer memory.

References

1. Brown D. Breaking RSA may be as difficult as factoring // Cryptology ePrint Archive. 2005. URL: https://eprint.iacr.org/2005/380.pdf
2. Aggarwal D., Maurer U. Breaking RSA generically is equivalent to factoring // Advances in Cryptology – EUROCRYPT 2009. 2009. URL: https://eprint.iacr.org/2008/260.pdf
3. Pomerance C., Lenstra H. W., Tijdeman R. Analysis and comparison of some integer factoring algorithms // Computational methods in number theory. 1982. Issue 1. P. 89–139.
4. Vasilenko O. N. Teoretiko-chislovye algoritmy v kriptografii. Moscow, 2003. 328 p.
5. Ishmuhametov Sh. T. Metody faktorizacii natural'nyh chisel: ucheb. pos. Kazan', 2011. 213 p.
6. Korneyko A. V., Zhilin A. V. Analiz izvestnyh vychislitel'nyh metodov faktorizacii mnogorazryadnyh chisel // Modeliuvannia ta i formatsiyni tekhnolohiyi. 2011. Issue 61. P. 3–13.
7. Howey E. Primality Testing and Factorization Methods // Semantic Scholar. 2014. URL: https://www.semanticscholar.org/paper/Primality-Testing-and-Factorization-Methods-Howey/7fd44eb2df7b39716e984d548c28f51d9dc6bbbb
8. Wu M.-E., Tso R., Sun H.-M. On the improvement of Fermat factorization using a continued fraction technique // Future Generation Computer Systems. 2014. Vol. 30. P. 162–168. doi: https://doi.org/10.1016/j.future.2013.06.008

9. Knuth D. Art of Computer Programming, Vol. 2. Seminumerical Algorithms. 3-rd Edition. Massachusetts, 1997. 762 p.
10. Bressoud D. Factorization and Primality Testing. Springer-Verlag, 1989. 237 p. doi: https://doi.org/10.1007/978-1-4612-4544-5
11. Burton D. Elementary Number Theory. 7-th Edition. TMG, 2012. 436 p.
12. Somsuk K., Tientanopajai K. An Improvement of Fermat's Factorization by Considering the Last m Digits of Modulus to Decrease Computation Time // International Journal of Network Security. 2017. Vol. 19, Issue 1. P. 99–111. doi: http://doi.org/10.6633/IJNS.201701.19(1).11
13. Somsuk K., Kasemvilas S. MFFV2 and MNQSV2: Improved Factorization Algorithms // 2013 International Conference on Information Science and Applications (ICISA). 2013. doi: https://doi.org/10.1109/icisa.2013.6579415
14. Somsuk K., Kasemvilas S. MFFV3: An Improved Integer Factorization Algorithm to Increase Computation Speed // Advanced Materials Research. 2014. Vol. 931-932. P. 1432–1436. doi: https://doi.org/10.4028/www.scientific.net/amr.931-932.1432
15. Usman M., Bajwa Z., Afza M. New Factoring Algorithm: Prime Factoring Algorithm // International Journal of Engineering and Management Research. 2015. Vol. 5, Issue 1. P. 75–77. URL: https://pdfs.semanticscholar.org///10d5/4bf2a4b8f46a99efa195077024d82212683.pdf
16. Somsuk K., Kasemvilas S. Possible Prime Modified Fermat Factorization: New Improved Integer Factorization to Decrease Computation Time for Breaking RSA // Advances in Intelligent Systems and Computing. 2014. P. 325–334. doi: https://doi.org/10.1007/978-3-319-06538-0_32
17. Somsuk K. A new modified integer factorization algorithm using integer modulo 20's technique // International Computer Science and Engineering Conference (ICSEC'14). 2014. P. 312–316. URL: https://www.semanticscholar.org/paper/A-new-modified-integer-factorization-algorithm-20's-Somsuk/d70a527c0a4a7c671c814aad6de140dcbabe4445
18. Wu M.-E., Tso R., Sun H.-M. On the Improvement of Fermat Factorization // Lecture Notes in Computer Science. 2012. P. 380–391. doi: https://doi.org/10.1007/978-3-642-34601-9_29
19. Vinnichuk S. D., Maksimenko E. V. Mnogokratnoe prorezhivanie dlya uskoreniya metoda faktorizacii Ferma pri neravnomernyh shagah dlya neizvestnoy // Visnyk NTUU "KPI". Informatyka, upravlinnia ta obchysliuvalna. 2016. Issue 64. P. 13–24.
20. Maksymenko Ye. Selection of effective basic basis of module with multiple thinning trial value in the factorization fermat's method with irregular pitch // Informatyka ta matematychni metody v modeliuvanni. 2016. Vol. 6, Issue 3. P. 270–279.
21. Vynnychuk S., Maksymenko Y. Formation of non-uniformity increment for the basic module base in the problem of fermat's factorization method // Information Technology and Security. 2016. Vol. 4, Issue 2. P. 244–254.