Lietuvos matematikos rinkinys
Proc. of the Lithuanian Mathematical Society, Ser. A
Vol. 64, 2023, pages 25–35
https://doi.org/10.15388/LMR.2023.33590

Vilnius University Press

Investigation of an attack on the multi-prime RSA cryptosystem based on cubic equations

Aleksėjus Michalkovič, Jokūbas Žitkevičius

Kaunas University of Technology, Department of Mathematics and Natural Sciences
Studentų str. 48, LT-51367 Kaunas, Lithuania
E-mail: aleksejus.michalkovic@ktu.lt; jokubas.zitkevicius@ktu.edu

Received July 7, 2023; published online November 20, 2023

Abstract. In this paper we consider a modification of the attack on the classic RSA cryptosystem aimed at factoring the public modulus n, which is a product of three primes. To improve the performance of the modified attack we introduce additional parameters. We present the theoretical upper bound on the search range parameter and define a shifting parameter based on the empirical results. Since these changes make our attack probabilistic, we investigate the dependence of the success on the values of the newly defined parameters.

Keywords: asymmetric cryptography; multi-prime RSA; integer factorization problem

AMS Subject Classification: 94A60

1 Introduction

The story of asymmetric cryptography began in 1976 when W. Diffie and M. Hellman published their groundbreaking paper [3]. Later in 1977 R. Rivest, A. Shamir, and L. Adleman introduced their cryptosystem based on operations in the ring of integers $\mathbb{Z}_n$, where n is a publicly known composite number obtained as a product of two large primes p and q [6]. This system later became known by the abbreviation RSA and to this day is used in practice for authorization, e-signature, etc.

Due to the popularity of this cryptosystem, many attacks on it have been presented. Two of the most popular ones were proposed by M. Wiener and D. Coppersmith [1]. These attacks are aimed at such weak spots of RSA as low private and public exponents respectively. Wiener's attack proposed in [7] uses continued

fractions to compute the low private exponent d satisfying the following identity:

$$d < \frac{1}{3} n^{\frac{1}{4}},$$

where $n = pq$ is a publicly known modulus. Coppersmith's attack described in [2] uses the LLL algorithm to recover the message μ encrypted by RSA using a small public exponent e. Such bad choices include $e = 3$ or $e = 17$, which were popular due to their binary representations.

These attacks demonstrated some potential weaknesses of RSA, which, unfortunately, result in a total break of the system. Nowadays, due to the gained knowledge, these weak spots can be avoided. However, such attacks create additional obstacles in the implementation of RSA.

The most dangerous attacks on RSA are aimed at factorizing the composite modulus n. As of now, the best attack is the general number field sieve, which is most effective for composites $n > 10^{110}$. The current record for the factorization of large composite is the 829-bit RSA modulus [8]. It is widely recommended that nowadays the primes p and q have to be at least 1024-bits long to ensure the cryptographic security of the classic RSA.

Evidently, these primes are enormous. Hence, modifications of the original cryptosystem were proposed. One such modification is the multi-prime RSA scheme which keeps all the ideas of the original except for the computation of the composite modulus n, which can now be a product of more than two primes. For simplicity, we limit ourselves to considering composites of the form $n = pqr$, where p, q, r are distinct primes. An advantage of this approach is that these primes can be smaller as compared to the original idea, yet the composite n remains a hard nut to crack. For example, the 2048-bit composite n can be computed as a product of three 683-bit primes which is the expected bit length of factors in this case. Another example of the modifications of RSA is the Okamoto-Uchiyama cryptosystem [5] which uses a public modulus of the form $n = p^2 q$. It is also worthy to mention that using brute-force attack it would take an infeasible amount of iterations to find the prime factors. On the other hand, the algorithm we present in this paper greatly relies on the additional information on the size of factors and is mainly aimed at saving computational time.

In this paper we consider an attack proposed in [4] aimed at factorizing the public modulus n. The idea of their proposal is based on solving a quadratic equation $x^2 - bx + n = 0$ with an unknown coefficient b using Vieta's theorem. We present the original idea in Section 2 and modify the presented attack to the case of multi-prime RSA when the public modulus n is a product of three primes in Section 3. In Section 4 we investigate the performance of the modified attack and present several examples. As usual, conclusions are presented at the end of the paper.

2 The original attack

Suppose that the public modulus $n = pq$ is known and our goal is to find the prime factors p and q. Consider the quadratic equation $x^2 - bx + n = 0$. Due to Vieta's theorem, we have $b = x_1 + x_2$, where x_1 and x_2 are the roots of the considered equation. Since we are interested in the integer solutions, the possible pairs up to

permutations are (p, q) and $(1, n)$, where $n = pq$. However, we have:

$$p + q < n + 1. \tag{1}$$

Hence we can define the iterative search of the true value of b by varying it starting from some initial value b_0. The search terminates at the first sensible solution pair hence minimizing the value of b. However, due to inequality (1) the solution pair corresponding to the minimal value of b is (p, q) as desired. The described iterative process works correctly given that the starting value b_0 is chosen properly. The explicit description of this process is presented below:

1. The starting value is obtained by assuming that $p \approx q$. Then we have $n \approx p^2$ and $b \approx 2p$. Hence, the starting value is chosen as an even integer closest to $2\sqrt{n}$;
2. In each iteration the value of the discriminant $b^2 - 4n$ using the current value of b is calculated;
3. If the value of the discriminant is a perfect square, we solve the considered quadratic equation and terminate the search, i.e. the factorization is found;
4. Otherwise we increase b by 2 and return to Step 2.

Note that the reasoning behind the correctness of the considered attack is Vieta's theorem and the minimization of the coefficient b without specifying if the found roots are actually primes. Due to the structure of n and the inequality (1) they have to be, given that we are looking for integer values. This may not be the case if n factors into more than two primes. Theoretically there is nothing that prevents this algorithm from producing a sensible solution for $n = pqr$. However, there is no guarantee that both integer roots x_1, x_2 are primes. This fact means that the relation between the considered attack and the Goldbach's conjecture is non-existent, despite the authors of [4] claiming otherwise.

Another problem of the presented attack is the fact that for the composite $n = pqr$, where p, q, r are equal in bit size, the roots x_1 and x_2 are far apart, i.e. x_2 is twice as long as x_1 in bits. This fact greatly complicates the search for this pair, since the number of iterations increases drastically.

For these reasons we introduce the modified version of the presented attack aimed at factoring a composite modulus of the form $n = pqr$, where p, q and r are three distinct primes.

3 The modification of the proposed attack

Suppose the public modulus n is known to be a product of three distinct primes p, q and r, and our goal is to find these factors. Similarly to the concept presented in [4], we can consider the following cubic equation:

$$x^3 - bx^2 + cx - n = 0, \tag{2}$$

where b, c, d are coefficients determined by *Vieta's theorem for a cubic equation whose roots are x_1, x_2 and x_3 in the following way:

$$b = x_1 + x_2 + x_3;$$
$$c = x_1x_2 + x_2x_3 + x_3x_1.$$

The main objective of the attack is to factor a composite number n by solving a cubic equation (2) whose coefficients are $b = p + q + r$, $c = pq + qr + rp$ and $n = pqr$. Then due to Vieta's formulas the primes p, q, and r are the roots of the considered cubic equation.

However, compared to the technique presented in [4] we have to vary two coefficients instead of one. This comes from the fact that the coefficient d is fixed to be equal to n whereas the other two coefficients have to be varied since the values of the prime factors of n are unknown.

Our first task is to determine the starting values b_0 and c_0 of the iterative process for the coefficients b and c respectively. To find the correct values of b and c, we have to start the iterative process using $b_0 \leqslant b$ and $c_0 \leqslant c$. Therefore, using AM-GM inequality we get the estimates as:

$$b = x_1 + x_2 + x_3 \geqslant 3\sqrt[3]{x_1x_2x_3} = 3\sqrt[3]{n};$$
$$c = x_1x_2 + x_2x_3 + x_3x_1 \geqslant 3\sqrt[3]{x_1^2x_2^2x_3^2} = 3\sqrt[3]{n^2}.$$

Since both b_0 and c_0 have to be integers, we round the obtained estimates using the ceiling function, i.e. $b_0 = 3\left\lceil \sqrt[3]{n} \right\rceil$ and $c_0 = 3\left\lceil \sqrt[3]{n^2} \right\rceil$. Moreover, since b_0 and c_0 have to be odd due to Vieta's theorem, we add 1 to these estimates, if any of them are even. We also keep b_i and c_i odd throughout the iterative search. Note that if $3\sqrt[3]{n}$ is already an integer, then n is a perfect cube and we are done since b_0 is the desired prime.

As mentioned above, we expect that each factor of 2048-bit composite n is approximately 683 bits long. Note, however, that a even the slightest change of bit length of prime factors may cause giant gaps between them, since 682-bit and 683-bit primes can differ by as much as 2^{682}. In this paper, we stick with this intuitive guess. However, we use much smaller composites to perform our experiments.

It is also important to note that theoretically the estimates are bounded above by the values $b_{\max} = n + 2$ and $c_{max} = 2n + 1$, which corresponds to the solution $x_1 = 1, x_2 = 1, x_3 = n$. However, for every fixed value b, reaching the theoretical maximum of the coefficient c is extremely time-consuming, therefore, we need to take into account few tendencies throughout the search. The numerical results, however, showed that in the very first iteration the ratio $\frac{c_0}{b_0}$ is nearly equal to the theoretical ratio $\frac{c}{b}$.Although, the difference of those ratios might not be positive, we considered taking absolute value of the difference of their ratios as an indicator of our problem difficulty. Namely, if the difference is small, then our algorithm finds the factors relatively quickly because the values that need to be checked are quite limited. On the other hand, if the difference is large, the algorithm takes much more time. However, we cannot determine the coefficients b, c and their ratio $\frac{c}{b}$. Therefore, we only take i-th iteration ratio $\frac{c_i}{b_i}$ into consideration and compare it to $\frac{c_0}{b_0}$ by taking the absolute value of their difference. This way, we introduce the following parameter:

$$\lambda_i = \frac{c_i}{b_i} - \sqrt[3]{n} \approx \frac{c_i}{b_i} - \frac{c_0}{b_0}, \tag{3}$$

where i is an iteration index.

Let us also denote the actual value of the previously defined difference as:

$$\lambda^* = \frac{c}{b} - \sqrt[3]{n}. \tag{4}$$

Let us assume that within the i-th iteration the following condition holds true:

$$|\lambda_i| \leqslant \lambda, \tag{5}$$

where $\lambda > 0$ is a parameter preset prior to the start of the search. The purpose of λ is to prevent the situation when the estimates are relatively far from target coefficients and bound the search space in order to find any meaningful solutions of (2). Note, however, that the values of b and c used in (3) are the correct set of coefficients, i.e. these coefficients are taken from the exact cubic equation (2) whose roots are the desired primes. Evidently, since the correct values of b and c are unknown, we have to guess the value of λ. However, this means that our attack becomes probabilistic, i.e. there is a chance that a factorization is not found since the right set of coefficients was not considered due to the λ limit. One of the major problems of our attack is determining a suitable value of λ to keep the right set of coefficients within range while also keeping the search as fast as possible.

Now we can describe our attack:

1. We compute b_0, c_0 and insert the estimates into (2);
2. Solve the cubic equation (2) and check if the solutions are integers. If they are, we have found the desired factors and we terminate the search;
3. If the factor were not found, we increase the value of c_i by two, i.e. $c_{i+1} := c_i + 2$ whereas $b_{i+1} := b_i$;
4. If the new pair of coefficients (b_{i+1}, c_{i+1}) satisfies the inequality (5) we continue the attack from Step 2;
5. Otherwise, we redefine our pair of parameters (b_{i+1}, c_{i+1}) as follows:

$$b_{i+1} := b_i + 2, c_{i+1} := \hat{c}_0, \tag{6}$$

where $\hat{c}_0 = \min(c_k) \in \{c_0, c_1, \dots\}$ is shifted to the minimal c_k so that the pair (b_{i+1}, c_{i+1}) satisfies the inequality (5) and continue the search from Step 2.

To prove the correctness of the presented attack let us note that due to (3) the desired primes are the roots of the cubic polynomial (2). Moreover, it can be easily shown that these primes minimize the values of coefficients b, c. However, since the presented algorithm terminates after finding the first sensible solution (x_1, x_2, x_3), it has to return a valid factorization of n, since the coefficients b, c are minimized. The parameter λ ensures that the iterative process never reaches other solutions, e.g. $(1, p, qr)$, etc. Hence, for the well-chosen parameter λ, we can conclude that the attack works correctly, i.e. it returns the prime factors of n.

4 Investigation of the proposed attack

Let us explore the dependence λ on the size of the primes and approximate the maximal value of this parameter required to find the primes with 100% probability at the cost of computational time.

We start by evaluating the minimum and maximum values of λ^*. Define the following target function:

$$\lambda^*(p, q, r) = \frac{pq + qr + rp}{p + q + r} - \sqrt[3]{pqr}. \tag{7}$$

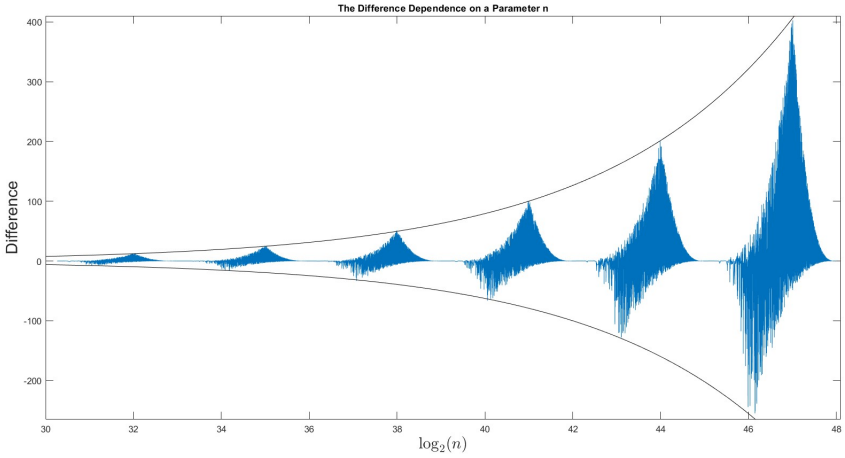

Fig. 1. The dependence of the parameter λ^* on the binary logarithm of n.

In this paper, we assume that each prime is m bits long to observe tendencies between the primes, i.e. we have $2^{m-1} < p, q, r < 2^m$. Hence we have the following symmetrical variable-wise constraints:

$$\begin{aligned} 2^{m-1} - p < 0, \quad p - 2^m < 0; \\ 2^{m-1} - q < 0, \quad q - 2^m < 0; \\ 2^{m-1} - r < 0, \quad r - 2^m < 0. \end{aligned}$$

Future work in this field may include the investigation of our attack for factors of non-equal bit lengths.

Using KKT conditions we get that the minimum of the target function (7) with the above constrains is at point $(2^m - 1, 2^{m-1} + 1, 2^{m-1} + 1)$ whereas its maximum is at point $(2^{m-1} + 1, 2^m - 1, 2^m - 1)$. Plugging these points into (7) we can show that $|\max(\lambda^*)| \geqslant |\min(\lambda^*)|$. However, to shorten this paper we omit these calculations. Furthermore, by ignoring the ones in the obtained maximum point we get the even larger value of λ^* and, therefore, the upper bound for our guess of the value of λ is:

$$\lambda < 2^m \Big(\frac{4}{5} - \frac{1}{\sqrt[3]{2}} \Big). \tag{8}$$

Then assuming that n factors into distinct 8-bit primes the upper bound of λ is approximately 1.61, whereas for 16-bit factors this bound is 412.84. Clearly, we can see the exponential growth of this bound and for primes used in practice the upper bound for λ is roughly 2.53×10^{203}, which approximates to $2^{675.7}$.

However, it is important to note that the upper bound (8) is purely theoretical and can be used to ensure that the factorization of n can be found given enough time. In practice this bound may be smaller resulting in faster execution of the considered attack.

For this reason we generate a relatively large amount of prime triplets (p, q, r), calculate $n = pqr$, λ^* and $\log_2(n)$ for each case. We sort the values of n in an ascending order, keeping relevant λ^* result. The graph presented in Fig. 1 describes the dependence of λ^* on the binary logarithm of n.

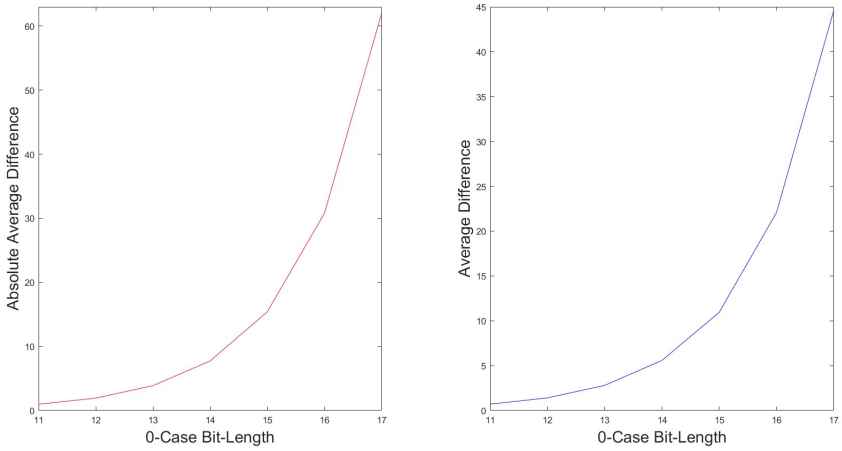

Fig. 2. The dependence of the mean and the absolute mean of λ^* on the bit size of each prime.

Taking every peak of positive values and every peak of negative values separately in Fig. 1, we can successfully use exponential regression to produce two exponential functions using those points with more than 98% compatibility. These functions are plotted as black curves joining the peaks.

We can see from Fig. 1 that λ^* depends on the distance between factors of n rather than on n itself, i.e. taking relatively close values of n_1 and n_2 we get drastically different λ_1^* and λ_2^*. Therefore, we cannot determine the value of λ fairly close to the real value λ^* knowing only the value of n.

We can also see from the presented graph that the value of λ^* peaks out quite rarely and for the most part its absolute value is significantly smaller than both extrema. For this reason we investigate the arithmetic mean value of λ^*. Also we present the mean of the absolute values of λ^*, which can help us to make a better prediction for the value of λ.

Note that in Fig. 2 all of the primes in the factorization of n are equal in their bit size. We refer to this case as '0-case bit-length'. In the future it may be a good idea to investigate primes of distinct sizes to inspect the effect it has on the execution of the considered attack.

Let us now present several examples of the considered attack. We keep track of the following data: the primes p, q, r equal in bit size in all of the examples, the parameter λ pre-chosen prior to the execution of the attack, the true value λ^* and the number of iterations needed to find these primes using the presented algorithm. This data is summarized in Table 1.

In the first two examples we can see that the guess λ was rather close to the true value λ^* and hence relatively small amount of iterations was needed to find the primes p, q, r. Also, note that despite the fact that the primes in the second example are larger, they are closer to each other as compared to the first one. For this reason we can see that the number of iterations differs roughly 6.5 times in favour of the second example.

Table 1. Examples of successful factorization of public modulus $n = pqr$.

p	q	r	λ	λ^*	Iterations
547	839	983	2	1,884	158912
31391	31957	32297	0,1	0.005	24163
26107	26573	27799	5	−0.079	4997711
31847	31873	32341	5	−0.003	285038

In the latter two examples we can see that λ was chosen poorly, i.e. the true value λ^* significantly differs from our guess. In the third example we can see the dire affect this poor choice had on the iteration number. Comparing second and third examples we see that despite the fact that in both cases the values of λ^* were close to zero and to each other in absolute value the number of iterations differs roughly 207 times in favour of the second example.

Notably in the fourth example despite the poor choice of λ we can see that the number of iterations was quite reasonable. However, this comes from the fact that the starting values of the coefficients b_0 and c_0 were relatively close to true values of b and c respectively. A better choice of λ would greatly decrease the number of iterations as demonstrated in the second example.

To increase the performance of the algorithm, we also modify Step 5 by introducing another parameter $\delta > 0$. The purpose of this modification is to optimize the iterative process by diminishing the search space of the coefficients.

To determine the value of δ, we generated $100'000$ prime triplets (p, q, r), where the bit size of each prime is m and considered the ratio $\frac{c-c_0}{2^m}$ for each triplet. Note that for this investigation we are interested only the first iteration. Below in Table 2 we present the dependence of the average of the considered ratio on the bit length of the factors.

Table 2. The dependence of the average space between c and c_0 on the primes bit length.

m	8	10	12	14	16	18
$\frac{c-c_0}{2^m}$	4.450	16.526	65.210	269.529	1082.491	4182.155

We can see that the considered ratio on average is proportional to the size of primes, i.e. we can assume that within every increase of bit for each prime, the considered ratio increases 2 times. Using the presented Table 2, let us define δ as a function of the prime bit length m in the following way:

$$\delta = \delta_0 \cdot 4.450 \cdot 2^m \cdot 2^{m-8} = \delta_0 \cdot 4.450 \cdot 2^{2m-8}, \tag{9}$$

where $0 < \delta_0 < 1$ is a shifting parameter. For more clarity, assume that the original search space of coefficients b, c is:

$$S = \{(b_i, c_j) \,|\, b_0 \leqslant b_i \leqslant b_{\max}, c_0 \leqslant c_j \leqslant c_{\max}\}.$$

The parameter δ reduces this search space to S' as follows:

$$S' = \{(b_i, c_j) \,|\, b_0 \leqslant b_i \leqslant b_{max}, c_0 + \delta \leqslant c_j \leqslant c_{\max}\}.$$

Evidently, considering the search space S', fewer iterations are performed which accelerates the primes search. On the downside, we are no longer able guarantee that these primes are found if the value of δ_0 is too large. We can only determine the probability of that outcome given that the parameter λ defines a large enough range of search, e.g. it was chosen using the upper bound (8)

To estimate the probability of success we generated relatively large number of prime triplets (p, q, r) of equal bit length m. Then we vary the value of δ_0 and calculate δ using its definition (9). In Table 3 we present the probability of successfully restoring m-bit factors of n when the search space is S'.

Table 3. The probability of a successful attack while choosing δ_0 for each of primes p, q, r in bit size of m.

m \ δ_0	0.1	0.3	0.5
10	0.7811	0.6054	0.5127
14	0.7726	0.6032	0.5105
18	0.7662	0.6147	0.5162

The presented Table 3 depicts the approximate tendency of the attack success and we can see a notable difference between reducing the search space by 10% of iterations as compared to 50% reduction which grants us approximately 77% and 51% probability of success respectively.

Note also that in practice the upper bound of λ is enormous and hence the total scan of the search space S is infeasible. Even if we greatly reduce this search space, it still remains too large for the total scan. On the other hand, to achieve searchable S' the value of δ_0 has to be close to 1. However, in this case the probability of successful attack is essentially 0.

To end this section we present several examples which demonstrate the dependence of the number of iterations on the parameter δ_0. The data is summarized in Table 4.

Table 4. Examples of successful factorization of public modulus $n = pqr$ with reduced search space.

p	q	r	λ	λ^*	δ_0	Iterations
563	761	967	3	0.367	0.0	178845
563	761	967	3	0.367	0.3	163611
563	761	967	3	0.367	0.5	148015
563	761	967	0.5	0.367	0.0	31228
563	761	967	0.5	0.367	0.3	28016
563	761	967	0.5	0.367	0.5	25168
2459	2957	3467	5	0,241	0.0	1835775
2459	2957	3467	5	0,241	0.3	1278174
2459	2957	3467	5	0,241	0.5	838678
2459	2957	3467	0.5	0,241	0.0	191932
2459	2957	3467	0.5	0,241	0.3	129286
2459	2957	3467	0.5	0,241	0.5	85411

We can see from the presented table that the number of iterations significantly decreases when δ_0 increases. Moreover, for the 36-bit composite n (the larger of the

considered composites) it seems that the true values of the coefficient c is somewhat close to the lower bound $c_0+\delta$. For this reason the change of the number of iterations between the values of $\delta_0 = 0.3$ and $\delta_0 = 0.5$ is more noticeable than in the other case. Note that, if we are using brute-force attack, finding the minimum prime p takes $\frac{p-1}{2}$ iterations, since we are searching for only odd numbers, whereas to find the other factor we need $\frac{\sqrt[3]{n}-p}{2}$ iterations at most. Using the standard length of primes, i.e. each of them is 683 bit, the brute-force method needs at most $\lfloor\frac{p-1}{2}\rfloor+\lfloor\frac{\sqrt[3]{n}-p}{2}\rfloor = \lfloor\frac{\sqrt[3]{n}-1}{2}\rfloor = \frac{\sqrt[3]{n}}{2}-1 = 2^{682}-1$ iterations. In respect to this result, the modified algorithm of RSA becomes probabilistic, depending on chosen parameters λ, δ, however, the choice of the parameters becomes the reason of varying number of iterations (not the bit length of primes p, q, r), therefore, this algorithm becomes much more valuable time-wise comparing to brute-force attack when the bit length of factors increases drastically.

5 Conclusions

In this paper we considered a modification of a previously proposed attack on the classic version of RSA cryptosystem. Comparing these attacks we can see that both of them rely on Vieta's formulas for second and third degree polynomials respectively. Moreover, we have demonstrated that the original version does not have any relation to the Goldbach's conjecture.

However, since in our case two unknown coefficients have to be varied we introduced additional parameters λ^* and δ defined by expressions (7) and (9) respectively. Using these parameters we improved the performance of the modified attack at the cost of turning it probabilistic.

Empirically we have shown that maximal value of λ^* increases exponentially as the bit length of the prime factors grows. We have also evaluated the theoretical upper bound for the maximal value of λ^*. The presented results have shown that for 2048-bit composite n this bound is roughly 2^{676} which is far too large to perform an effective iterative search. Moreover, the exponential growth also applies to the negative values of λ^*.

We have also considered the average value of λ^*. Comparing graphs presented in Figs. 2 and 1 we can see that the average growth of λ^* is significantly slower than its maximal value. Hence we can reduce the range parameter λ while the success of attack remains reasonably high.

Since the peaks of negative values of λ^* are always less than the peaks of positive values, we can introduce a shifting parameter δ. This way we diminish the searching space and hence reduce the number of iterations. However, the probability of success noticeably decreases.

References

[1] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Not. Am. Math. Soc.*, **46**(2):203–213, 1999.

[2] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptol.*, **10**(4):233–260, 1997.

[3] W. Diffie, M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, **22**(6):644–654, 1976. https://doi.org/10.1109/TIT.1976.1055638.

[4] C. Liu, C.-C. Chang, Z.-P. Wu, S.-L. Ye. A study of relationship between RSA public key cryptosystem and Goldbach's conjecture properties. *Int. J. Netw. Secur.*, **17**(4):445–453, 2015.

[5] T. Okamoto, U. Shigenori. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology—EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Proceedings 17*, pp. 308–318, Finland, May 31–June 4, 1998. Springer.

[6] R.L. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, **21**(2):120–126, 1978.

[7] M.J. Wiener. Cryptanalysis of short rsa secret exponents. *IEEE Trans. Inf. Theory*, **36**(3):553–558, 1990. https://doi.org/10.1109/18.54902.

[8] P. Zimmermann. Factorization of RSA-250. Technical report, INRIA, Nancy, France, 02 2020.

REZIUMĖ

Trijų pirminių RSA kriptosistemos atakos, paremtos kubinėmis lygtimis, tyrimas

A. Michalkovič and J. Žitkevičius

Šiame darbe nagrinėjame atakos prieš klasikinę RSA kriptosistemą modifikaciją, kuria siekiama išskaidyti pirminiais daugikliais viešąjį modulį n, kuris yra trijų pirminių skaičių sandauga. Norėdami pagerinti modifikuotos atakos greitaveiką mes įvedame papildomus parametrus. Pateikiame teorinę viršutinę paieškos diapazono parametro ribą ir apibrėžiame poslinkio parametrą pagal empirinius rezultatus. Kadangi šie pakeitimai daro mūsų ataką tikimybinę, mes tiriame atakos sėkmės priklausomybę nuo naujai apibrėžtų parametrų.

Raktiniai žodžiai: asimetrinė kriptografija; RSA kriptosistema; skaidymo pirminiais daugikliais uždavinys