RECENT ADVANCES IN INFORMATION TECHNOLOGY

EDITED BY WALDEMAR WÓJCIK AND JAN SIKORA

Faculty of Electrical Engineering and Computer Science, Lublin University of Technology, Lublin, Poland

LUBLIN 2017

CONTENS

PREFACE

In this book, entitled: " RECENT ADVANCES IN INFORMATION TECHNOLOGY "contains a body of work whose common denominator is a modern information technology, without which it is difficult to imagine progress in modern technologies and manufacturing processes.

Topics addressed in the work is very extensive ranging from the game theory to the advanced control problems.

The individual chapters are separate, closed issues, which are the fruit of the cooperation of Ukrainian Kazakh and Polish scientists.

The editors and the authors hope that this issue will allow to strengthen friendship between our communities and consequently between our nations.

Waldemar Wójcik

Jan Sikora

Lublin, July 2017

CHAPTER 1

Security estimates updating of Asymmetric cryptosystems for modern computing

V.K. Zadiraka , A.M. Kudin, I.V. Shvidchenko & P.V. Seliukh,
V.M. Glushkov Institute of Cybernetic of NAS of Ukraine, Kyiv, Ukraine

P. Komada,
Lublin University of Technology, Lublin, Poland

A. Kalizhanova,
al-Farabi Kazakh National University, Almaty, Kazakhstan

INTRODUCTION

Today RSA is still one of the most common and widespread asymmetric cryptosystems. A lot of cryptographic protocols that deal with encryption, digital signature, and distribution of key information are based on RSA scheme. The basis of scheme's security is a complex theoretical and numerical factorization problem: to find the prime divisors of large *n*.

The issue of cryptosystem's security in practice is associated with the task of correct choice of cryptosystem parameters. A recent research by a group of scientists led by Arjen Lenstra (2012) confirms the problem of RSA implementation, namely generation of parameters that will ensure the system's security in practice. From among 11.5 million of RSA key certificates under research there has been found more than 26000 vulnerable keys with size of 1024 bits and 10 keys with size of 2048 bits. In this case the vulnerability was to factorize the cryptosystem's module.

This vulnerability was implemented through the use of common divisors for modules, what is more, only a small amount of such common divisors was found as a result of key reissuing for the same owner. At most this situation is caused by the presence of the global problem of generating "qualitative" modules that are built with large prime numbers (Bernstein et al. 2013, Heninger et al. 2012), besides, the authors conclude that crossing to larger modules does not lead to the expected decrease of vulnerable keys number. The mechanisms for generating prime numbers that are used for constructing modules should be additionally studied.

Given the above we formulate the problems of effectiveness analysis of applying different computational models (including probabilistic, parallel and hybrid) for factorization and discover the existence of polynomial complexity algorithm of numbers factorization of special form; and as a result to clarify practical security of RSA cryptosystem.

We will discover the amount of "secure" RSA modules. If we find out the limited amount of such modules of defined size and this limitation turns out to be the polynomial of the module's size then there is a critical issue of RSA security in practice, namely if the cryptanalysis or RSA is P (by factoring the module).

Adi Shamir offered the research of RSA module break "cost" of certain length by constructing a specific factorization machine (Shamir 1999). This research was continued in Shamir, Tromer and Bernstein papers (Shamir et al. 2003, Bernstein 2001). This task is urgent now due to cloud computing and flexible computing architecture models (Programmable logic device (PLD)) rise. To research the impact of new computer (cloud computing, etc.) technology on complexity and RSA module break cost is the main idea of the article.

1.1. THE EFFECTIVENESS OF DIFFERENT COMPUTATIONAL MODELS FOR NUMBERS FACTORIZATION

Let us briefly look through the characteristics and applicability of sequential, parallel and probabilistic computational models. It should be noted that quantum calculations and their applicability to the cryptography problems, in particular factoring problem have gained a wide interest in recent years (Zadiraka et al. 2013). P.W. Shor (1999) has proved the polynomial complexity of factorization and discrete logarithm for this computational model. This gives an additional possibility to construct effective factor algorithm and to discover more precisely the connection between computational models and their practical implementation. This topic requires a separate discussion, so it is beyond the scope of this chapter.

The most studied sequential computational model uses determined operations performed one after another and the results of the previous calculations are used in the next step. Deterministic factoring algorithms are fully expressed through this model of computations. Today there are no practical ways to significantly improve the performance of such factoring algorithms for arbitrary input.

The extension for sequential model is probabilistic. It is a good framework for inherently probabilistic algorithms. There are two types of algorithms: Monte Carlo and Las Vegas. The algorithms of the first type give the answer that may be correct for some random sequences that are generated during the algorithm, but may be incorrect. The probability of false answers is reduced by repeated execution of the algorithm.

The Las Vegas algorithms respond correctly or finish with answer "don't know". The probability of getting an incorrect result is zero. Exactly the study of probabilistic factoring algorithms can lead to getting an effective performance of algorithms for numbers factoring of special form. This can happen due to the "nature" of Las Vegas algorithm to "split" input data by subsets with different estimates of algorithm's complexity.

Consider the algorithm which can be divided into blocks of operations, the result of which indirectly affects the further work, so the branches for independent computing that can be executed simultaneously are allowed. This arrangement of computing defines the parallel computational model. The application of parallel model for k separated branches is expected to give at best the k time's speedup of algorithm runtime (linear decrease computational complexity).

Unfortunately, in practice, to achieve such speedup for factoring algorithm is impossible because of the inability to efficiently use CPU time of each of k branches through the existing need for communication between processors.

The well-known Brent's review (Brent 1990) states that certain factor algorithms ECM (Elliptic Curve Method (Lenstra 1987) for example) are good-suited for parallelization that yields almost linear effect of decreasing computational complexity, but some algorithms (including rho-Pollard method (Pollard 1975)) can achieve this effect only in theory (Crandall 1999).

Recent research of Pollard's method (Crandall 1999, Koundinya 2013) shows that the linear effect can be achieved only for the small number of parallel branches. Note that there are two possible ways to parallelize the algorithm – to build various elements of one random sequence, or to look up different sequences. Namely, the application of the second approach and possibility of its effective improvement are discussed below.

Note that the mentioned above can be applied only for general factoring algorithms. It is clear that these estimates are the upper bounds for factorization the numbers of special form. Moreover, the study the effectiveness of factorization algorithms determines the so-called problem of "impact of algorithm optimization on the PC's structure". Or rather it can be defined as the problem of constructing an effective computational model (possibly hybrid) for number factoring and discovering its implementation on the base of programmable logic devices PLD, GPUs. This confirms the relevance of discovering the existence of time-efficient factoring algorithms of special numbers, some results are discussed below.

1.2. RHO-POLLARD ALGORITHM

Pollard's rho-method is a probabilistic algorithm that once made a significant breakthrough and still is not completely researched. The complexity of the algorithm does not depend on the factor but on its divisors. Thus, like the ECM algorithm (Lenstra 1987), Pollard's method is effective for factoring multi-prime RSA modules, such numbers of fixed length that are the product of more than two primes.

Pollard used two simple ideas: the birthday paradox (Sekey 1993) and the ability to check divisibility of number n by p computing $GDC(x-y,n)$, if $x \equiv y \bmod p$. We can choose the starting point x_0, function f and construct recurrent sequence $x_i = f(x_{i-1})$ which plays the role of a "random" sequence. Considering it by the module p, where p is a divisor of n there must exist two comparable elements.

Pollard chose $f(x) = x^2 + 1$ as a recurrence. The subtraction of two elements is always split into a product $x_k - x_j = x_{k-1}{}^2 - x_{j-1}{}^2 = (x_{k-1} - x_{j-1})(x_{k-1} + x_{j-1})$. These divisors are tested as possible for the number n.

This brings up the question – is it possible to obtain a significant improvement in the performance of the algorithm by choosing another recurrent function. The study of this issue (Hartman 1982) was carried out only with the use of numerous assumptions that narrow the set of numbers to factor. Despite the application of different functions the effectiveness turned out to be negative. So we considered the sets of numbers reduced by the module p and the capacity of such sets for the following cases: $p = 2p' + 1$, where p' is prime; $p = 3 (\bmod 4)$ or $p = 2 (\bmod 3)$.

Hartman obtained the following results about the capacity of the sets that are generated by different functions.

1. $\{(ax+b) \bmod p, x = 0,1,..., p-1\}$ is complete set of residues. It is determined as a permutation.
2. If *k* is odd and $(k, p-1) = 1$, then $\{x^k \pmod p, x = 0,1,..., p-1\}$ is a permutation.
3. If $P(x) \pmod p$ is a permutation, then $(aP(x)+b) \pmod p$ is a permutation for $a \neq 0 \bmod p$.
4. If $P(x) \pmod p$ generates the set of size n, then $(aP(x)+b) \pmod p$ and $P(ax+b) \pmod p$ are the different sets, but also of the size n.
5. $P(x) = x^2 \pmod p$ generates the set of size $\frac{p+1}{2}$. This statement is correct for all polynomials of degree 2.
6. If $P_1(x), P_2(x)$ are permutations, then $P_1(x), P_2(x)$ does not generate the whole set of residue. Little is known about the size of the set.
7. Cubic polynomial $P(x) = ax^3 + 3bx^2 + 3cx + d$ separates the complete set of residues into two size-equivalent subsets corresponding to the following:
   a. x^3 of size $p(b^2 = ac)$;
   b. $x^3 + x, x^3 - x$ of size $2\frac{p+1}{3} - 1 (b^2 \neq ac)$.
8. $P_1(P_2(x))$ generates the set of size less than or equal $\min(P_1(x), P_2(x))$.

But to discuss only the size of set does not make sense. It is important to consider how much calculations should be done to find an item that is congruent to module p with the previously calculated one. If the selected function, and therefore, the generated sequence has many "small" loops and does not have "long" loops and "tails" then the probability to find quickly the decomposition of number n is high (Fig. 1.1). But hypothetically this is possible only under the condition that n has only small prime divisors. If sequence has "long" loops then the probability that the selected starting element will lead us to the desired collision is high and the probability of choosing a starting element, which quickly leads to collision, is low.

To use high-degree functions is impractical because of the large time-costs of sequence calculations. Using the linear function $f(x) = ax + b \bmod N$ the subtraction $x_k - x_j = (x_0 + kc) - (x_0 + jc)$ does not seem to be random comparing with $f(x) = x^2 + 1$ proposed by Pollard (1975).

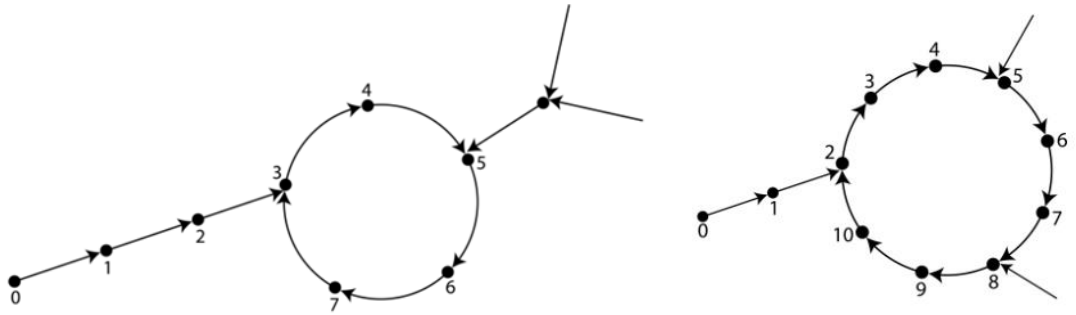

Fig. 1.1. Loops and "tails" in rho-Pollard method

Applying two different linear recurring sequences also does not give any improvements in performance; we can conclude nothing about the residue of elements of two such sequences $x_k - x_j = kc_1 - jc_2$, where c_1, c_2 are the constants. As we can see, the matter of selecting the function that generates random sequence for Pollard's rho-method is still open.

1.3. SELECTING CRYPTO PARAMETERS

Let us define n as a module and e as an exponent for RSA cryptosystem; these parameters are the public key. Cryptosystem's module is a product of odd primes p_i, $i = 1,2,..,u$ where $u \geq 2$, and public exponent e is an integer that takes the value between 3 and $n-1$, $GCD(e, \lambda(n)) = 1$, where $\lambda(n) = GCD(p_1 - 1,..., p_u - 1)$ – generalized Euler function and some additional conditions (Mukhachev et al. 2005). Private key d is a positive integer that satisfies $e \cdot d \equiv 1 (\bmod \lambda(n))$.

Let consider RSA key size as a length of module n in bits. NIST limits the set of key size values as 1024, 2048 and 3072 bits. In addition, restrictions are imposed on the generation of module n only as a product of two primes $n = p \cdot q$ (FIPS PUB 186-4-1).

The Public-Key Cryptography Standards (PKCS) #1 of version 2.1 and more it is allowed to use modules that are the product of more than two numbers (multi-prime modules), but it does not determine any restrictions on the choice of divisors. For a fixed size of the module the usage of more than two divisors reduces the size of such divisors and thus increases the probability of these modules to be factored by the algorithm, the complexity of which depends not on the size of factored number but on the size of number's divisor.

To deploy RSA cryptosystem first we need to select exponent e, fixed or random integer $2^{16} < e < 2^{256}$. It is not permitted to choose random primes p and q to generate module with size of 1024 bits, but only those that satisfy the following (FIPS PUB 186-4-1):

- $p-1$ has prime divisor p_1,
- $p+1$ has prime divisor p_2,
- $q-1$ has prime divisor q_1,
- $q+1$ has prime divisor q_2.

Here p_1, p_2, q_1, q_2 are so-called auxiliary primes, p i q are primes with conditions. For modules of size 2048 and 3072 bits we may use random primes p, q, that are built using probabilistic or constructive algorithms of generating prime numbers (Maurer 1990, Gordon 1984, Zadiraka et al. 2007, Shawe-Taylor 1986). Table 1.1 shows the limitation on the length of numbers when using different algorithms of generating primes. The function $len(p)$ is the bit length of p.

Table 1.1. Valid length values for auxiliary primes p_1, p_2, q_1, q_2

$len(p)$	**Min. length of auxiliary primes**	**Max. value of** $len(p_1)+len(p_2)$	
		p, q **probable primes**	p, q **provable primes**
1024	>100 bits	<496 bits	<239 bits
2048	>140 bits	<1007 bits	<494 bits
3072	>170 bits	<1518 bits	<750 bits

Probably the prime number p is a number generated by the following algorithm: to determine the length of the desired number p; to generate odd random number; to check if the number is prime using a probabilistic test. The most widespread is Miller-Rabin test. The algorithm tests if given N is prime following the next steps:

1. To calculate t and s so that $N-1=2^t s$, s has to be odd.
2. To randomly choose b, $2 \le b \le N-2$.
3. To calculate $y = b^s \bmod N$.
4. If $y \neq 1$ and $y \neq N-1$, then while $i < t$ and $y \neq N-1$ starting from $i=1$ do: $y = y^2 \bmod N$. If $y = 1$, then Exit with "N is composite", otherwise $i = i+1$.
5. If $y \neq N-1$, then Exit with "N is composite".
6. Exit with "N is composite".

While constructing the module the following requirements on the amount of rounds of Miller-Rabin testing of the generated number should be met. This is caused by the decreasing probability to choose composite number as a prime with each additional round. Below there is the table (see table 1.2) with the amount of required rounds of testing for error level less than 2^{-100} is given.

Table 1.2. Minimum number of rounds of Miller-Rabin testing when generating primes for RSA scheme, the error level is 2^{-100}

Parameters	Number of rounds
$p_1, p_2, q_1, q_2 > 100$ bits p and q : 512 bits	For p_1, p_2, q_1, q_2 : 38 For p i q : 7
$p_1, p_2, q_1, q_2 > 140$ bits p and q : 1024 bits	For p_1, p_2, q_1, q_2 : 32 For p i q : 4
$p_1, p_2, q_1, q_2 > 170$ bits p and q : 1536 bits	For p_1, p_2, q_1, q_2 : 27 For p i q : 3

Note that the use of probably prime numbers while implementing the cryptosystem not only significantly increases the time of module constructing but in case of mistaken use of composite number can cause malfunctions, which reduces the resistance to cryptanalysis. So we refer to deterministic algorithms that construct prime numbers in polynomial time. With the help of these algorithms we can reduce the problem of selecting the module by testing number's primality to the problem of generating provably prime number. Let's take a look at Maurer's (1990) and Shawe-Taylor's algorithms (1986). They are the so-called "tornado" algorithms: small prime number is used at the first iteration. Each following iteration determines provably prime number in a certain range that guarantees an increase of the resulting number.

Maurer's algorithm of generation of provably prime random number

The input of the algorithm is integer k – the number of bits of the desired prime. The algorithm uses the following parameters: L is the border for trial divisions; M is a parameter that ensures the existence of the desired prime. $M = 20$ is recommended.

1. If $2^k < L$, then to generate random odd k-bit number N and to test its primality with trial division. Repeat until a prime number N is generated. Exit.
2. If $k \le 2M$, then let $r = \frac{1}{2}$, else repeat the following until $k - rk > M$.

   2.1.1. Choose random integer s, $0 \le s \le 1$, $r = 2^{s-1}$.
3. $k_1 = [rk] + 1$. Repeat steps 1-2 to construct k_1-bit random prime q.
4. Let $t = 2^{k-1}/(2q)$.
5. Let R is random integer, $t < R \le 2t$. $N = 2Rq + 1$.
6. Let a is integer and $1 < a < N - 1$. If $a^{N-1} (\mathrm{mod}\, N) = 1$ and $GCD(a^{2R} - 1, N) = 1$, then Exit with "N is prime".
7. Else repeat steps 5-6.

The second algorithm of constructing provably prime random number is Shawe-Taylor's algorithm. The input of the algorithm is an integer k – the number of bits of the desired prime. The algorithm uses parameter L as a border for trial divisions.

Shawe-Taylor's algorithm

1. If $2^k < L$, then to generate random odd k-bit number N and to test its primality with trial division. Repeat until a prime number N is generated. Exit.

2. If k is odd, let $k_1 = (k+3)/2$. If it is even, let $k_1 = k/2+1$. Recursive pass the algorithm with input parameter k_1 to construct a k_1-bit prime number.
3. Let x is random integer and $2^{k-1} \le x < 2^k$.
4. Let t is the smallest integer greater than $x/(2q)$.
5. If $2tq+1 \ge 2^k$, then let t be the smallest integer greater than $2^{k-1}/(2q)$.
6. Let $N = 2tq+1$.
7. Chose random integer a, $1 < a < N-1$, let $x = a^{2t} \bmod N$. If $x \ne 1$, $GCD(x-1, N) = 1$, $x^q = 1 \bmod N$, then N is generated prime.
8. Else let $t = t+1$ and repeat steps 5-7.

However, the above algorithms guarantee only the primality of generated numbers. Additional requirements for numbers that are divisors of RSA module are the property of their so-called "maximization of the complexity of specialized factoring algorithms" (Zadiraka et al. 2007). For example, for $p \pm 1$ algorithms this property is revealed if n has prime factor p so that number $p \pm 1$ has sufficiently large power-smooth boundary (Mukhachev and Khoroshko 2005), in other words, $p \pm 1$ has large prime factors. This divisor p is called "strong" prime number. So we bring up an important question about the number and the distribution of such strong primes.

1.4. DISTRIBUTION AND AMOUNT OF STRONG PRIMES

Let p be a classic strong prime if the following requirements are met:

$$p = 1(\bmod r),\ p = -1(\bmod s),\ r = 1(\bmod\ t), \tag{1.1}$$

where r, s, t are large primes. This means that p, r, s, t can be shown as $p = 2jr+1$, $p = 2ks-1$, $r = 2lt+1$, whereby the lower the numbers j, k, l the better.

Note that a generalization of the classical notion of prime number is a concept of strong primes that maximize the complexity of all known algorithms for factorization.

R.L. Rivest denied the idea of need to use strong primes as factors for RSA modules (Rivest and Silverman 1999). On the base of Lenstra's algorithm he wanted to show that the choice of p did not take an effect on the efficiency of obtaining the factorization, and that, in general, it is enough to choose random primes to obtain RSA modules. Later his ideas were refuted, since some algorithms, including ECM, work better if the number $p-1$ is smooth.

Here follows the Gordon's method of generating strong prime numbers.

1. Construct a random prime number s of pre-selected size. We can select a pseudorandom number x of the desired size and using trial divisions we may leave the numbers in the range $[x, x+\log_2 x]$ that do not have small divisors. Among the remaining numbers choose prime number s with the help of primality test.
2. Generate random t in the same way.
3. Using trial division and primality test generate prime $r = 2lt+1$, sorting out l in $[1, \log_2 t]$.
4. Calculate $u = u(r,s) = (s^{r-1} - r^{s-1}) \bmod rs$.
5. If u is odd, then let $p_0 = u$, else $p_0 = u + rs$.
6. Test if $p = p_0 + 2krs$ is prime for $k = 0,\ 1,\ 2,\ ...$

Assume the condition of applying strong primes while generating cryptosystem's module. The following conditions for factors should be met:

$$\begin{aligned} &p_1 \mid p-1, p_1 \geq n^{x_1}, p_2 \mid p_1-1, p_2 \geq n^{x_2}, p_3 \mid p+1, p_3 \geq n^{x_3}, 0<x_1, x_2, x_3<0.5 \\ &q_1 \mid q-1, q_1 \geq n^{x_4}, q_2 \mid q_1-1, q_2 \geq n^{x_5}, q_3 \mid q+1, q_3 \geq n^{x_6}, 0<x_4, x_5, x_6<0.5 \end{aligned} \tag{1.2}$$

Let $p_i(n)$ be i-th the greatest divisor of n. Let $w_i(n,x) = \#\{t : 1 \leq t \leq n, p_i(t) \leq n^x\}$ is the amount of integers less or equal n for which $p_i(n)$ does not exceed n^x. Applying the result of Knuth and Trabb Pardo (1976) we have the probability that random number has the greatest divisor bigger than some border:

$$P\{q_1 > n^x\} = \int_x^1 \frac{dt}{t} = -\ln x. \tag{1.3}$$

Using Adamar and Vallee Poussin (Porter 1915) estimates we obtain the following estimate of the number of classic strong primes in $[a,b]$:

$$\pi_s(a,b) = -\ln x_1 \cdot \ln x_2 \cdot \ln x_3 \cdot \int_a^b \frac{dt}{\ln t}. \tag{1.4}$$

Considering the independence of the described divisors the estimate amount of RSA modules is defined as:

$$(\pi_s(a,b))^2. \tag{1.5}$$

Let us compare the resulting estimate of the number of RSA modules generated with random primes (no requirements for "strong" primes). RSA module is a so-called semiprime number because it has exactly two prime factors. The number whose prime factors less particular bound will be called B-smooth. Works (Ishmukhametov and Sharifullina 2014) present the distribution of semiprime and smooth numbers, the estimates are based on Riemann hypothesis. Probabilistic function $g(y)$ corresponds the probability that the number y will be "semiprime". Function approximation using series summation by prime numbers p:

$$g(y) \approx \sum_{p \leq \sqrt{y}} \frac{1}{p(\ln y - \ln p)}. \tag{1.6}$$

Using Mertens's formula:

$$\sum_{p<x} \frac{\ln p}{p} = \ln x + O(1), \tag{1.7}$$

and Abel's theorem results we have:

$$g(y) \approx \frac{\ln\ln y - \beta}{\ln y}, \text{ where } \beta = 1 + \ln\ln 2. \tag{1.8}$$

Let $\psi(x, y)$ be the amount of all y-smooth numbers less than x, we have the recurrent formula for the calculation:

$$\psi(x, p_k) \approx \sum_{i=0}^{t_k} \psi(\frac{x}{p_k^i}, p_{k-1}),\ t_k = \left[\frac{\ln x}{\ln p_k}\right],\ k > 1, \tag{1.9}$$

where p_k – k-th prime ($p_1 = 2$), $\psi(x, 2) = \left[\frac{\ln x}{\ln 2}\right] + 1.$

There are rough estimates for calculating the number of smooth numbers because with the growth of smoothness bounds the computational complexity increases exponentially. Hildebrand (1986) obtained the following estimation:

$$\psi(x, y) \approx x \cdot \rho(u)\left\{1 + O\left(\frac{\ln(u+1)}{\ln y}\right)\right\},\ \text{where } x = y^u. \tag{1.10}$$

Here $\rho(u)$ is the Dickman-de Bruijn function, $u = \ln x / \ln y$, $\rho(u)$ is the solution of differential equation $u\rho'(u) + \rho(u-1) = 0$ for $u > 1$, this approximation is valid only with the simultaneous growth of x and y keeping $\ln x / \ln y$ as a constant. Here is the approximate formula for $y < (\ln x \ln\ln x)^{1/2}$:

$$\psi(x, y) \approx \frac{1}{\pi(y)!} \prod\left(\frac{\ln x}{\ln p}\right)\left\{1 + O\left(\frac{y^2}{\ln x \ln y}\right)\right\}. \tag{1.11}$$

But to estimate the capacity of the set of smooth numbers with practically used boundaries this approximation is inapplicable.

The considered estimations of semiprime number's distribution give an opportunity to present the size of the whole set. To solve the problem of RSA crypto parameters choice we should restrict this set only by choosing such semiprimes that are not smooth for some smoothness border B in order to maximize the complexity of solving the factorization problem of semiprime numbers.

In the above-mentioned notations we are interested in the following estimation for the number of semiprimes in the range $[2, T]$ that are not B-smooth (Ishmukhametov and Sharifullina 2014):

$$K(T, B) = \int_2^T \frac{\ln\ln x - \beta}{\ln x} dx - \psi(T, B)\text{, where } \beta = 1 + \ln\ln 2. \tag{1.12}$$

On the other hand it is possible to evaluate the amount of B-smooth numbers in $[2, T]$ with the help of combinatorial methods and group theory. It makes sense to choose primes as smoothness boundary. So having fixed $B = p_k$ as k-th prime number, the problem to evaluate the number of combinations $x = \prod_{i=1,k} p_i^{\alpha_i}$ where $x < T$. Note that the estimates (1.4) and (1.12) are close in value, but (1.4) is obtained in much easier way and takes into account the condition of choosing "strong" primes.

In general, the following statements are applicable for implementing RSA cryptosystems:

1. it is necessary to impose the restrictions for primes that are used for generation RSA modules for system's security in practice;
2. the output set algorithms that generate such primes have appreciably less cardinality of a set than cardinality of prime number set.

For these reasons, we formulate the hypothesis that there exists a polynomial complexity algorithm that enumerates the set of "secure" RSA modules. The above estimations support this hypothesis.

1.5. FEATURES OF FACTORIZATION ALGORITHM ELABORATION FOR CLOUD COMPUTING

The first problem of factorization price estimating for cloud computing is a form of ordering the services, which are typically given by providers. For example, Windows Azure provides services in a form of virtual machine with specific characteristics of the CPU, memory and disk space (Accessed June 3 2016. http://azure.microsoft.com/en-us/pricing/details/cloud-services). The most powerful architecture which is offered "for working applications with large databases, server applications and high-speed applications" (e.g., "D14" has the characteristics of 112 GB RAM, 800 GB of disk space, 16 core processor) costs $2.611 US per hour or about $1943 US per month.

The cost of computing module rent with random configuration for most cloud platforms may be estimated using the on-line calculators (Accessed June 3 2016. https://cloud.google.com/ products/ calculator, http://www.hpcloud.com/pricing). Thus, a computer module in the cloud Google Cloud Platform with the configuration similar to "D14" costs $5.36 US per hour, but the monthly rent will cost $689.08 (Accessed June 3 2016. http://azure.microsoft.com/en-us/pricing/details/cloud-services). In HP's Public cloud the configuration, aimed at calculating the maximum power (30 HP Cloud Compute Units, 4 virtual cores, 60GB RAM, 540GB ephemeral disk) is $1.35/hr ($985.50/mo.) (Accessed June 3 2016. http://www.hpcloud.com/pricing). You can also choose virtually unlimited (theoretically – only for the price) number of such machines.

So, to order a specific computing architecture targeted at particular factorization algorithm is difficult, so you need to rely on many computing nodes of standard minimum configuration aimed at powerful calculations that operate in parallel. The price of this node (e.g., n1-highcpu-2 for the cloud Google Cloud Platform) is $38.84 per month.

The second problem is that for factorization price estimating it is also necessary to choose the fastest at present, universal (which can be used for any kind of numbers) algorithm, which allows the simple implementation of a parallel computing model. The choice of this algorithm for factorization price estimating has similar value to the choice of the universal models of computation (Turing-Post machine, normal Markovian algorithm, brute-force program, etc.) for algorithms complexity estimate.

The generally accepted solution to this problem is the choice of algorithm GNFS (Buhler et al. 1993, Lenstra and Lenstra 1993, Couveignes 1993, Ishmukhametov and Sharifullina 2014). This algorithm, like the most effective factorization algorithms at present time operates quite a simple idea, proposed by Fermat, such as: if you find a pair of numbers (A,B), that satisfy the equation $A^2 - B^2 = n$, then $n = (A-B)(A+B)$. So, if $m = \lfloor \sqrt{n} \rfloor$, then $x = 1,2,...$ calculate the value of polynomial $q(x) = (m+x)^2 - n$ until $q(x)$ is not a perfect square.

A simple generalization of Fermat's idea is to search for pairs of numbers that satisfy more general equation $A^2 - B^2 \equiv 0 \bmod(n)$ (Ishmukhametov and Sharifullina 2014). Along with this, it appears that the value of the polynomial $q(x)$ is the so-called "smooth" numbers. A pair of integers (A,B), is called "smooth" (on base factor F, formed of primes) if:

1. The relation $A^2 - B^2 \equiv 0 \bmod(n)$ is performed.

2. B is multiplied only by prime factors that belong to the set F.

Representation of polynomial values $q(x)$ by the elements of a vector space over F to search the full squares has led to Dixon algorithm first, then to quadratic sieve (QS), but as a generalization of the latter – the general number field sieve (GNFS). We will briefly present the main stages of this algorithm to discuss major evaluations of its performance (Ishmukhametov and Sharifullina 2014):

1. Choose irreducible polynomial degree $d \geq 3$.
2. We choose an integer $m, [n^{1/(d+1)}] < m < [n^{1/(d)}]$, and represent *n* on the basis of *m*:

$$n = m^d + a_{d-1}m^{d-1} + ... + a_0$$

3. With this representation we bind irreducible in the ring $Z[x]$ (the ring of polynomials of the variable x with integer coefficients) polynomial:

$$f_1(x) = x^d + a_{d-1}x^{d-1} + ... + a_0 .$$

4. Define sieving polynomial $F_1(a,b)$ as a homogeneous polynomial of two variables a and b:

$$F_1(a,b) = b^d * f_1\left(\frac{a}{b}\right)$$

5. Also define another polynomial $f_2(x) = x - m$ and the corresponding homogeneous polynomial $F_2(a,b) = a - bm$. The main requirement for pair selection (f_1, f_2) is to fulfill the condition: $f_1(m) = f_2(m) \bmod(n)$ which is obviously performed in our case, because the first polynomial at the point m is n, and the second is zero.
6. Choose two positive numbers L_1 and L_2, which define the certain rectangular area $SR = \{1 \leq b \leq L_1, -L_2 \leq a \leq L_2\}$, called "sieve area".
7. Let θ – root of polynomial $f_1(x)$. Consider the polynomial ring $Z[\theta]$ (practically root θ is not evaluated, and is only used for formal description of the algorithm). Define the algebraic factor base FB_1, consisting of a first-order polynomial form $a - b\theta$ with the norm which is prime number. These polynomials are simple irreducibility elements in the ring of algebraic field $K = Q[\theta]$. The absolute value of the norms of polynomials from factor base FB_1 are bound above by some constant B_1.
8. At the same time let define rational factor base FB_2, consisting of all prime numbers bound above by second constant B_2.
9. To be able to check the final stage of the algorithm, whether the determined polynomial is perfect square, we define a relatively small set of polynomials $c - d\theta$ of the 1 order, the norm of which is also a simple number. Let this set be marked as FB_3. It must satisfy the condition $FB_1 \cap FB_3 = \varnothing$ is called Quadratic Character Factor Base.

10. Further simultaneous sieving of polynomials based on FB_1 is performed and integers based on FB_2 to obtain the set M, consisting of smooth pairs (a,b). The pair (a,b) is called smooth if $GCD(a,b)=1$ and the polynomial $a-b\theta$ and the number $a-bm$ are multiplied entirely by the relevant factor bases FB_1 and FB_2. The number of smooth pairs in the set M must exceed the total power of three factor bases at least by two units.
11. The next step is to search the subset $S \subseteq M$ so that the product of all pairs $(a-b\theta)$ is H^2 and pairs $(a-bm)$ is B^2, $H \in Z, B \in Z$. To search the set S as in the quadratic sieve method, a system of linear algebraic equations is formulated with coefficients from of the set $F_2=\{0,1\}$, the solution of which will be the numbers of the set S.
12. Next we form the polynomial:

$$g(\theta)=(f_1'(x))^2\prod_{(a,b)\in S}(a-b\theta).$$

13. If the whole procedure is performed correctly, then the polynomial $g(\theta)$ is a perfect square in the ring of polynomials $Z[\theta]$. We extract square roots of the polynomial $g(\theta)$ and an integer B^2, finding some polynomial $\alpha(\theta)$ and the number B.
14. Replace the polynomial $\alpha(\theta)$ with the number $\alpha(m)$.
15. So, a pair of integers (A,B), satisfying the condition $A^2-B^2 \equiv 0 \bmod(n)$ has been found.

As seen from the description of the main stages of the algorithm there are two key parameters that affect its complexity – the size of the sieve area and the size of factor base. The complexity estimate of the algorithm GNFS is easy to measure by using $L_n(\alpha,c)=e^{\left(c+o(1)\right)\left((\ln n)^{\alpha}(\ln\ln n)^{1-\alpha}\right)}$ function. The most famous current estimates for the classical method GNFS – $L_n\left(\frac{1}{3};1{,}92..+o(1)\right)$ (Ishmukhametov and Sharifullina 2014), for the "group" GNFS (several numbers are factorized at a time for which common screening results, are used the average factorization time per each number is calculated) $L_n\left(\frac{1}{3};1{,}704+o(1)\right)$ (Bernstein and Lange 2014).

For approximate real cost estimate of factorization for the clouds given above let's give the estimate of the sieving area (the most complicated stage) of the classical method GNFS. It is $L_n\left(\frac{1}{3};\left(\frac{8}{9}\right)^{1/3}+o(1)\right)$.

If we take this estimate for the number of computing nodes of "minimal" configuration then the factorization cost estimate of 1024-bit number is hundred trillion US dollars, if the number is factorized during a month, which of course is much worse than the factorization cost estimates specially designed by using the devices for factorization (TWIRL or SHARK) (Shamir and Tromer 2003, Franke 2005). So the cost of the system SHARK (Franke 2005) for factorization of 1024-bit number is evaluated 160 million US dollars, if the number is factorized during a year.

Such estimate is explained first of all, by the lack of flexibility of cloud computing services order, and, secondly, by the direct use of existing implementations of the algorithm GNFS for cloud technologies. The doubtless advantage of cloud computing that determines the relevance of

further research in this area, is their relatively greater availability compared to specialized calculators and opportunity to use the so-called “peak-mobilization” operation mode in “critical” cases – the use of cloud resources across the corporation, state and international association.

1.6. CONCLUSIONS

Existing standards of implementation and usage of asymmetric cryptosystems impose restrictions on the crypto parameters generation. This situation is conditioned by modern advances in solving the problem of factorization of large numbers. The interest of assessing RSA security in practice is the highlighted issue of size or rather the nature of size growth of the set of secure crypto parameters, namely semiprime numbers that have additional constraints about their divisors.

REFERENCES

Baydenko P.V., Kudin A.M., 2012: *Efektyvnist' zastosuvannya alhorytmiv faktoryzatsiyi do moduliv kryptosystemy RSA*, X Vseukrayins'ka naukovo-praktychna konferentsiya studentiv, aspirantiv ta molodykh vchenykh “Teoretychni i prykladni problemy fizyky, matematyky ta informatyky”, Zbirka tez dopovidey uchasnykiv, Kyiv, Ukraine, pp. 253-254, (in Ukrainian).

Bernstein D.J., 2001: Circuits for integer factorization: a proposal, http://cr.yp.to/papers.html#nfscircuit.

Bernstein D.J., Chang Y.-A., Cheng C.-M., Chou L.-P., Heninger N., Lange T., van Someren N, 2013: *Factoring RSA keys from certified smart cards: Coppersmith in the wild*, Cryptology ePrint Archive, Report 2013/599, https://eprint.iacr.org/2013/598.pdf

Bernstein D.J., Lange T., 2014: *Batch NFS*, Selected Areas in Cryptography, LNCS, Springer, vol.8781, pp. 38-58, http://cr.yp.to/factorization/batchnfs-20141109.pdf.

Brent R.P., 1990: Parallel algorithms for integer factorization, Number Theory and Cryptography (edited by J.H. Loxton), London Mathematical Society Lecture Notes (Book 154), Cambridge University Press, pp. 26-37.

Buhler J.P., Hendrik W., Lenstra Jr., Pomerance C., 1993: *Factoring integers with the number field sieve*, The Development of the Number Field Sieve, Springer, vol.1554, pp. 50-94.

Couveignes J., 1993: *A general number sieve implementation*, Lecture Notes in Math., vol. 1554, pp. 103-126.

Crandall R.E., 1999: Parallelization of Pollard-rho factorization, http://academic.reed.edu/physics/faculty/crandall/papers/parrho.pdf.

Franke J., Kleinjung T., Paar C., Pelzl J., Priplata C., Stahlke C., 2005: *SHARK: a realizable special hardware sieving device for factoring 1024-bit integers*, Cryptographic hardware and embedded systems, LNCS, Springer, vol. 3659, pp. 119-130.

Gordon J., 1984: *Strong RSA keys*, Electronics letters, vol. 20(12), pp. 514-516.

Hartman W.J., 1982: *An Experimental Study of Monte Carlo Factoring Techniques*, National Telecommunications and Information Administration.

Heninger N., Durumeric Z., Wustrow E., Halderman J.A., 2012: *Mining Your Ps and Qs: detection of widespread weak keys in network devices*, In Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, pp. 205-220.

Hildebrand A., 1986: *On the number of positive integers* $\leq x$ *and free of prime factors* $> y$, Journal of Number Theory, vol. 22(3), pp. 289-307.

Ishmukhametov Sh.T., Sharifullina F.F., 2014: *On distribution of semiprime numbers*, Russian Mathematics, vol.58(8), pp. 43-48.

Koundinya A.K., Harish G., Srinath N.K., Raghavendra G.E., Pramod Y.V., Sandeep R., Kumar P.G., 2013: *Performance analysis of parallel Pollard’s RHO factoring algorithm*, International Journal of Computer Science & Information Technology (IJCSIT), vol. 5(2), pp. 157-164.

Knuth, Donald E., et al.: Selected Papers on Computer Languages, Stanford, CA, CSLI, 2003. ISBN 1-57586-382-0) (typewritten draft, August 1976).

Lenstra Jr. H.W., 1987: *Factoring integers with elliptic curves*, Annals of Mathematics, Second Series, vol. 126(3), pp. 649-673.

Lenstra A.K., Lenstra H.W., 1993: *The development of Number Field Sieve*, LNM, Springer-Verlag, vol.1554.

Lenstra A.K., Hughes J.P., Augier M., Bos J.W., Kleinjung T., Wachter C., 2012: Ron was wrong, Whit is right, Cryptology ePrint Archive, Report 2012/064, http://eprint.iacr.org/2012/064.pdf.

Maurer U.M., 1990: *Fast generation of secure RSA-moduli with almost maximal diversity*, Advances in Cryptology – EUROCRYPT '89, LNCS, Spinger-Verlag, vol. 434, pp.636-647.

Mukhachev V.A., Khoroshko V.A., 2005: *Metody prakticheskoy kriptografii*, Poligraf-Konsalting, Kyiv, Ukraine, (in Ukrainian)

Pollard J.M., 1975: *A Monte Carlo method for factorization*, BIT Numerical Mathematics, vol. 15(3), pp. 331-334.

Porter, M. B. 1915. Review: Cours d'Analyse Infinitésmale, by Ch.-J. de la Vallée Poussin. Bull. Amer. Math. Soc. 22. pp. 77–85.

Rivest R.L., Silverman R.D., 1999: *Are* "*strong*" *primes needed for RSA?* RSA Laboratories Seminar Series, Seminars Proceedings.

Sekey G., 1993: Paradoksy v teorii veroyatnostey i matematicheskoy statistike, Mir, Moscow, Russia (in Russian).

Shamir A., 1999: *Factoring large numbers with the TWINKLE device*, Cryptographic Hardware and Embedded Systems, LNCS, Springer-Verlag, vol. 1717, pp. 2-12.

Shamir A., Tromer E., 2003: *Factoring large numbers with the TWIRL device*, Advances in Cryptology – CRYPTO 2003, LNCS, Springer-Verlag, vol. 2729, pp. 1-26.

Shawe-Taylor J., 1986: *Generating strong primes*, Electronic letters, vol. 22(16), pp.875-877.

Shor P.W., 1999: *Polynomial Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM Review, vol. 41(2), pp. 303-332.

Zadiraka V.K., Kudin A.M., Oleksyuk A.S., 2007: *Adaptivnye algoritmy polucheniya prostykh chisel i ikh primenenie v kriptografii*, Komp'yuternaya matematika, no. 1, pp. 54-61, (in Ukrainian).

Zadiraka V.K., Kudin A.M., 2013: *Cloud computing in cryptography and steganography*, Cybernetics and Systems Analysis, vol.49(4), pp. 584-588.

Digital signature standard, National Institute of Standards and Technology, Federal Information Processing Standards Publication, FIPS PUB 186-4-1.

Google Cloud Platform pricing calculator, https://cloud.google.com/products/calculator.

HP Cloud pricing, http://www.hpcloud.com/pricing.

Microsoft Azure. Cloud services pricing, http://azure.microsoft.com/en-us/pricing/details/cloud-services.