Tatra Mt. Math. Publ. **33** (2006), 189–198

TATRA MOUNTAINS
Mathematical Publications

ANALYSIS OF ALGORITHMS OF GENERATING KEY PARAMETERS FOR THE XTR CRYPTOSYSTEMS

Maciej Grześkowiak

ABSTRACT. In the paper [A. K. Lenstra and E. R. Verheul: *The XTR public key system*, Proceedings of CRYPTO '00, Lecture Notes in Comput. Sci, 1880, Springer-Verlag, Berlin, (2000), pp. 1–19] a new public key system XTR was introduced. In the same paper they proposed a practical method to select primes p and q such that $q \mid (p^2-p+1)$, $p \equiv 2 \pmod 3$ which are the key parameters for XTR. An analysis of running times of algorithms generating such primes relies upon some unproved conjectures. The aim of the present paper is to analyze algorithms proposed by A. K. Lenstra and E. R. Verheul under the assumption of hypotheses H and S.

1. Introduction

The XTR public key system introduced by A. Lenstra and E. Verheul in 2000 [7] is based on a new method for representing elements of a subgroup of the multiplicative group of a finite field. XTR uses a subgroup $\langle g\rangle \subset \mathbb{F}_{p^6}$, known as *the XTR subgroup* of the prime order q. This group is a subgroup of subgroup of order p^2-p+1 of the field $\mathbb{F}_{p^6}$, where $p \equiv 2 \pmod 3$ is a prime number. Primes p, q and the trace of element $g \in \mathbb{F}_{p^6}$ over $\mathbb{F}_{p^2}$ are key parameters for the XTR public key system. Primes p and q have to be selected in such a way that $q > 6$ divides p^2-p+1, where $p \equiv 2 \pmod 3$. From the security point of view p and q should be sufficiently large for the discrete logarithm problem in the multiplicative group of the field $\mathbb{F}_{p^6}$ and $\langle g\rangle$ to be intractable.

In [7] two algorithms were proposed which find primes p and q as above, but analysis of these is not given there. We denote by $P(a,q)$ the least prime

2000 Mathematics Subject Classification: 11Y16.
Keywords: XTR public key cryptosystem, the least prime in an arithmetic progression, hypothesis H, primality testing.

The work described in this paper has been supported in part by a Foundation for Polish Science scholarship and by the State Committee for Scientific Research under Grant 0T00A00323.

$p \equiv a \pmod{q}$ for any $a, q \in \mathbb{N}$, $1 \leqslant a \leqslant q$, $(a,q) = 1$. Both algorithms require selection of prime $p \equiv a \pmod{q}$ for suitably chosen a and q. A well known theorem by Heath-Brown [5] gives $P(a,q) \ll q^{5.5}$. From this we cannot exclude the possibility that the number of steps we need to find such a prime is exponential. Moreover, let $f(x) = x^2 - x + 1 \in \mathbb{Q}[X]$, then one of the algorithms requires finding an integer r such that $f(r)$ is prime. It is a natural theoretical question, "How quickly can we find such a prime?" It is even not known if there exist infinitely many prime values of $f(x)$. Based on some plausible conjectures, we can show that algorithms proposed by Lenstra and Verheul are asymptotically polynomial-time algorithms.

In this paper we analyze running times of algorithms which generate key parameters p and q for the XTR cryptosystem assuming two conjectures: Hypothesis H and Hypothesis S. The former is related to primes which are values of irreducible polynomials in one variable with integer coefficients and positive leading coefficient. The latter concerns the smallest prime in arithmetic progression. Under the above assumptions we prove that selection of parameters p and q is possible in polynomial time.

This paper is organized as follows. For the reader's convenience we present Hypotheses H and S in Section 2. In Section 3 we recall algorithms given in [7] and we formulate our main theorems.

2. Hypothesis H and Hypothesis S

2.1 Hypothesis H

In 1958 Schinzel and Sierpiński proposed a conjecture concerning prime values of irreducible polynomials in one variable with integer coefficients and positive leading coefficient [13]. It is known as *Hypothesis H*. P. T. Bateman and R. A. Horn [2] gave a quantitative form of it.

Hypothesis 2.1. (*H*) *Suppose $f_1, f_2, \ldots, f_k \in \mathbb{Z}[x]$ are polynomials with positive leading coefficients of degrees $h_1, h_2, \ldots, h_k$ respectively, and $f(n) = f_1(n) f_2(n) \cdots f_k(n)$ has no fixed divisor greater than 1. Suppose that polynomials are irreducible over the field of rational numbers and no two of them are identical. Let $Q_N(f_1, f_2, \ldots, f_k)$ denote the number of positive integers n between 1 and N such that all numbers $f_1(n), f_2(n), \ldots, f_k(n)$ are prime. Then as $N \to \infty$ we have*

$$Q_N(f_1, \ldots, f_k) = \frac{C(f_1, \ldots, f_k)}{h_1 \ldots h_k} \int_2^N \frac{du}{\log^k u} + o\left(\int_2^N \frac{du}{\log^k u} \right), \tag{2.1}$$

where

$$C(f_1,\dots,f_k)=\prod_p\left\{\left(1-\frac{1}{p}\right)^{-k}\left(1-\frac{\rho(p)}{p}\right)\right\},$$

the product being taken over all primes and $\rho(p)$ *being the number of solutions of the congruence*

$$f_1(n)\dots f_k(n)\equiv 0 \pmod p.$$

Note that when $\rho(p)<p$ *for all* p, *the constant* $C(f_1,\dots,f_k)$ *is positive* [3] *and so* (2),*1 can then be written as*

$$Q_N(f_1,\dots,f_k)\sim\frac{C(f_1,\dots,f_k)}{h_1\dots h_k}\int\limits_2^N\frac{du}{\log^k u}. \tag{2.2}$$

2.2 Hypothesis S

In 1944 Linnik [8] proved that $P(a,q)\ll q^L$ for some absolute constant L. Various authors have estimated L, and the best published result to date is $L=5.5$, due to D. R. Heath–Brown [5]. Furthermore Titchmarsh [14] has observed that if the Generalized Riemann Hypothesis holds, then $P(a,q)\ll \big(\varphi(q)\big)^2(\log q)^4$. Schinzel and Sierpiński [13] and Kanold [6] conjectured that $P(a,q)\ll q^2$. In this paper we work with the following hypothesis, which was suggested in [4] and supported by a simple heuristic argument and numerical computation [15]. We denote it briefly by S.

Hypothesis 2.2. (S)

$$\max_{(a,q)=1} P(a,q)\ll q\log^2 q.$$

3. Algorithms of generating parameters p and q

In this section we recall algorithms which generate key parameters p and q for the XTR cryptosystem. The algorithms presented here are from [7]. For the purposes of this paper they have been written in a particular pseudocode. We explain the meaning of functions which will be used in these algorithms below.

The function $Random(r)$: returns a random integer r which fulfils conditions described after ':'. Let $n_1,n_2,\dots,n_r$ be positive integers. The function $IsPrimeQuick(n_1,n_2,\dots,n_r)$ returns `true` if every $n_1,n_2,\dots,n_r$ satisfies probabilistic test for primality or some primality test which is deterministic under the assumption of some hypotheses and both run in polynomial time. Otherwise function $IsPrimeQuick(n_1, n_2, \ldots, n_r)$ returns `false`.

A good example for probabilistic primality test which works in polynomial time is the Rabin test [12]. For every odd number n, the probability that n is a prime after repeating k independent iterations of the Rabin algorithm is at least $1 - \frac{1}{4^k}$. On the other hand, assuming Extended Riemann Hypothesis there exists a deterministic algorithm for primality testing which runs in polynomial-time $O(\log^5 n)$ bit operations [10]. This algorithm is known as the Miller test.

The function $IsPrime(n_1, n_2, \ldots, n_r)$ returns `true` if every $n_1, n_2, \ldots, n_r$ satisfy some deterministic primality test which runs in polynomial time, and `false` if otherwise.

It is known that in 2002 M. Agrawal, K. Kayal and N. Saxena [1] have discovered an unconditional polynomial-time deterministic algorithm for primality testing. They have given a deterministic $\tilde{O}(\log^{12} n)$ time algorithm for testing if a number n is prime or not, where symbol $\tilde{O}(t(n))$ denote $O\big(t(n)\mathrm{poly}(\log t(n))\big)$ and $t(n)$ is some function of n. We will denote by $\mathcal{PT}$ number of bit operations necessary to carry out deterministic primality test. For simplicity assume that $\mathcal{PT} \gg \log^3 n$.

ALGORITHM 3.1. (*selection of p and q*)

`do`
    `do`
        $Random(r)$: $r \in \left[\frac{1}{2}N^{\frac{1}{2}} + \frac{1}{2},\ N^{\frac{1}{2}}\right]$, $r \in \mathbb{N}$, $r \not\equiv 2 \pmod 3$;
        q `:=` $r^2 - r + 1$;
    `while not` $IsPrimeQuick(q)$;
`while not` $IsPrime(q)$;
r_0 `:=` $q\big(2 - (r \bmod 3)\big) + r$;
k `:=` 0;
`do`
    `do`
        p `:=` $3qk + r_0$;
        k `:=` $k + 1$;
    `while not` $IsPrimeQuick(p)$;
`while not` $IsPrime(p)$;
`return` (p, q).

THEOREM 3.2. *Assume Hypothesis (H). There exist constants b_0 and N_0 such that for every integer $N \geqslant N_0$ and an arbitrary real $\lambda \geqslant 1$, Algorithm*

(3.1) *in steps 1–6 finds an integer* r *and a prime* q *such that*

$$q = r^2 - r + 1\,, \quad \frac{N}{4} \leqslant q \leqslant N\,, \quad r \not\equiv 2 \pmod 3\,,$$

with probability greater or equal to $1 - e^{-\lambda}$ *after repeating* $[b_0 \lambda \log N]$ *steps of the algorithm. Every step 1–6 of the algorithm takes no more than* $\mathcal{PT}$ *bits operations.*

P r o o f. Since $f(x) = x^2 - x + 1 \in \mathbb{Q}[x]$ is irreducible in $\mathbb{Q}[x]$ and $\big(F(0), F(1)\big) = 1$, $f(x)$ satisfies assumptions of Hypothesis H. We conclude from (2.1) that there exists an integer N_0 that for every integer $N \geqslant N_0$

$$\begin{aligned} Q_{N^{\frac12}}(f) &= \big|\{r : 1 \leqslant r \leqslant N^{\frac12},\ f(r) = q\}\big| \\ &= \frac{C(f)}{2} \int_2^{N^{\frac12}} \frac{du}{\log u} + o\left(\int_2^{N^{\frac12}} \frac{du}{\log u}\right), \end{aligned} \tag{3.1}$$

where q is a prime and

$$C(f) = \prod_p \left\{ \left(1 - \frac{1}{p}\right)^{-1} \left(1 - \frac{\rho(p)}{p}\right) \right\}, \tag{3.2}$$

the product (3.2) being taken over all primes and $\rho(p)$ being the number of solutions of the congruence

$$x^2 - x + 1 \equiv 0 \pmod p\,.$$

Since

$$\int_2^x \frac{du}{\log^k u} \sim \frac{x}{\log^k x}\,,$$

for every integer k, as can easily be shown via integration by parts, and using (2.2), (3.1) we obtain

$$\left| \left\{ r : \frac12 N^{\frac12} + \frac12 \leqslant r \leqslant N^{\frac12},\ f(r) = q \right\} \right| \sim C_0(f) \frac{N^{\frac12}}{\log N}\,,$$

where $C_0(f) = C(f)2^{-1}$. We denote by A_r the event that a random integer $r \in \big[\frac12 N^{\frac12} + \frac12,\ N^{\frac12}\big]$ is such that $f(r)$ is a prime. Hence the probability that in k trials A_r does not occur is

$$\begin{aligned} \left(1 - \frac{C_0(f)}{\log N}\right)^k &= \exp\left(k \log\left(1 - \frac{C_0(f)}{\log N}\right)\right) \\ &\leqslant \exp\left(\frac{-C_0(f)k}{\log N}\right) \leqslant e^{-\lambda} \end{aligned}$$

for an arbitrary real $\lambda \geqslant 1$ and $k = b_0 \lambda \log N$, where $b_0 = C_0(f)^{-1}$. Then the probability that in k trials A_r does occur is greater or equal to $1 - e^{-\lambda}$. From the above it follows that after repeating $[b_0 \lambda \log N]$ steps $1-6$ with probability greater than or equal to $1 - e^{-\lambda}$ the algorithm finds an integer r and a prime $q = f(r)$. Now, we estimate the number of bit operations required to carry out the steps $1-6$. It takes a fixed period of time to generate a random bit, and $O(\log N)$ a fixed number of bit operations to generate a random integer $r \in \left[\frac{1}{2}N^{\frac{1}{2}} + \frac{1}{2}, N^{\frac{1}{2}}\right]$. Computation $r^2 - r + 1$ can be done with $O(\log^2 N)$ bit operations. The most time-consuming step of the algorithm is the deterministic primality test for a number q which takes no more than $\mathcal{PT}$ operations. $\square$

Theorem 3.3. *Assume the Hypothesis (S). If r_0 and $3q$ are coprime integers, then Algorithm 3.1 in steps $9-14$ finds a prime p such that*

$$p \equiv r_0 \pmod{3q}, \quad q \leqslant p \ll q \log^2 q .$$

It can be computed after repeating $O(\log^2 q)$ steps of the algorithm. Every step $9 - 14$ of the algorithm takes no more than $\mathcal{PT}$ bit operations.

P r o o f. According to Dirichlet's theorem the arithmetic progression r_0: $(\bmod\, 3q)$ contains infinitely many primes. By Hypothesis S we have

$$\max_{(r_0,3q)=1} P(r_0, 3q) \ll q \log^2 q .$$

It follows immediately that we should perform not more than $O(\log^2 q)$ steps of the algorithm in order to find a prime p such that $q \leqslant p \ll q \log^2 q$. Moreover, computation of a number p can be done with $O(\log^2 q)$ bit operations and deterministic primality test for p takes no more than $\mathcal{PT}$ bit operations. $\square$

Corollary 3.4. *Assume the Hypotheses (H) and (S). There exist such constants b_1 and N_1 that for every integer $N \geqslant N_1$ and an arbitrary real $\lambda \geqslant 1$ Algorithm 3.1 generates such primes p and q, key parameters for the XTR cryptosystem, that*

$$\frac{N}{4} \leqslant q \leqslant N, \quad q \leqslant p \ll N \log^2 N,$$

with probability greater or equal to $1 - e^{-\lambda}$ after repeating $\left[b_1(\lambda + \log N)\log N\right]$ loops of the algorithm. Every step of the algorithm takes no more than $\mathcal{PT}$ bit operations.

P r o o f. Let r be an integer, $r \in \left[\frac{1}{2}N^{\frac{1}{2}} + \frac{1}{2},\ N^{\frac{1}{2}}\right]$. Assume that $q = r^2 - r + 1$ and $p \equiv r \pmod q$, $p \equiv 2 \pmod 3$ where p and q are primes. We see at once that $p^2 - p + 1 \equiv 0 \pmod q$. If $r \not\equiv 2 \pmod 3$, then $(3, q) = 1$ and from *The Chinese Remainder Theorem* we obtain

$$\begin{aligned} p &\equiv q + r \pmod{3q} \text{ for } r \equiv 1 \pmod 3, \\ p &\equiv 2q + r \pmod{3q} \text{ for } r \equiv 0 \pmod 3. \end{aligned} \tag{3.3}$$

It is easily seen that the algorithm in loop $1-6$ generates a prime number $q = r^2 - r + 1$ and in loop $10-14$ finds a prime p given by (3.3). Then the algorithm generates key parameters for the XTR public key system.

We now estimate complexity of the algorithm. Note that $(3q, r_0) = 1$ where $r_0 = q + r$, $r \equiv 1 \pmod 3$ or $r_0 = 2q + r$, $r \equiv 0 \pmod 3$. Then q and r_0 satisfy the assumptions of Theorem 3.3. Hence in loop $10-14$ we find a prime p after repeating no more than $O(\log^2 N)$ loop of the algorithm. Now we can conclude from Theorems 3.2 and 3.3 that there exist such constants N_1 and b_1 that for every integer $N \geqslant N_1$ and arbitrary real $\lambda \geqslant 1$ we find a prime q with probability greater or equal than $1 - e^{-\lambda}$ and a prime p after repeating no more than $[b_1(\lambda + \log N) \log N]$ loops of the algorithm. Every step of the algorithm takes no more than $\mathcal{PT}$ bit operations. □

ALGORITHM 3.5. (*selection of p and q*)

1. `do`
2.   `do`
3.   *Random*$(q) \in [N, 2N]$, $q \equiv 7 \pmod{12}$;
4.   `while not` *IsPrime Quick*(q);
5. `while not` *IsPrime*(q);
6. *Finds roots* r_i, $i = 1, 2$, *of polynomial* $X^2 - X + 1 \pmod q$;
7. *Finds roots* $r_i^0 := q\big(2 - (r_i \mod 3)\big) + r_i$;
8. $k := 0$;
9. `do`
10.   `do`
11. p `:=` $3qk + r_i^0$;
12. k `:=` $k + 1$;
13.   `while not` *IsPrimeQuick*(p);
14. `while not` *IsPrime*(p);
15. `return` (p, q).

THEOREM 3.6. *There exist such constants d_0 and N_0 that for every integer $N \geqslant N_0$ and an arbitrary real $\lambda \geqslant 1$ Algorithm 3.5 in steps $1-5$ finds a prime $q \equiv 7 \pmod{12}$, $N \leqslant q \leqslant 2N$, with probability greater or equal to $1 - e^{-\lambda}$ after repeating $[d_0 \lambda \log N]$ steps of the algorithm. Every step $1-5$ of the algorithm takes no more than $\mathcal{PT}$ bits operations.*

P r o o f. We denote by $\pi(x, q, a)$, $x \geqslant 1$, $a, q \in N$, $1 \leqslant a \leqslant q$, $(a, q) = 1$, the number of primes $p \leqslant x$ in the arithmetic progression $a, a + q, a + 2q, \ldots$

Then there exists such a constant N_0 that for every integer $N \geqslant N_0$ [11],

$$\pi(N,12,7) = \big(1+o(1)\big)\frac{1}{\varphi(12)}\,\frac{N}{\log N}\,, \tag{3.4}$$

where φ is the Euler totient function. Moreover,

$$\big|\{q : N \leqslant q \leqslant 2N,\ q \equiv 7 \pmod{12},\ q\text{—prime}\}\big| \sim \frac{N}{4\log N}\,.$$

We denote by A_q the event that a random integer $q \in [N, 2N]$, $q \equiv 7 \pmod{12}$ is prime. Hence the probability that in k trials A_q does no occur is

$$\begin{aligned}\left(1-\frac{1}{4\log N}\right)^k &= \exp\left(k\log\left(1-\frac{1}{4\log N}\right)\right)\\ &\leqslant \exp\left(\frac{-k}{4\log N}\right) \leqslant e^{-\lambda}\end{aligned}$$

for any an arbitrary real $\lambda \geqslant 1$ and $k = d_0\lambda\log N$ where $d_0 = 4$. Then the probability that in k trials A_q does occur is greater than or equal to $1-e^{-\lambda}$. It is obvious that the most time-consuming step of the algorithm is deterministic primality test which takes no more than $\mathcal{PT}$ bit operations. Then we conclude that with probability greater or equal to $1-e^{-\lambda}$ one can find a prime $q \equiv 7 \pmod{12}$ after execute no more than $[d_0\lambda\log N]$ steps of the algorithm. □

Theorem 3.7. *Assume Hypothesis (S). If $(r_i^0, 3q) = 1$ and $r_i \equiv 2 \pmod 3$ is not a prime for $i = 1$ or $i = 2$, then Algorithm 3.5 in steps 9–14 finds a prime p such that*

$$p \equiv r_i^0 \pmod{3q}\,, \qquad q \leqslant p \ll q\log^2 q\,.$$

It can be computed after repeating $O(\log^2 q)$ steps of the algorithm. Every step 9–14 of the algorithm takes no more than $\mathcal{PT}$ bit operations.

P r o o f. The proof is analogous to Theorem 3.3. □

Corollary 3.8. *Assume the Hypothesis (S). If $r_i \equiv 2 \pmod 3$ is not a prime for $i = 1$ or $i = 2$, then there exist such constants d_1 and N_1 that for every integer $N \geqslant N_1$ and an arbitrary real $\lambda \geqslant 1$ Algorithm 3.5 generates such primes p and q, key parameters for the XTR cryptosystem, that*

$$N \leqslant q \leqslant 2N, \quad q \leqslant p \ll N\log^2 N\,,$$

with probability greater than or equal to $1-e^{-\lambda}$ after repeating $\big[d_1(\lambda + \log N)\log N\big]$ loops of the algorithm. Every step of the algorithm takes no more than $\mathcal{PT}$ bit operations.

P r o o f. We first show that p and q are key parameters for the XTR public key system. Suppose that

$$q \in [N, 2N]\,, \quad q \equiv 7 \pmod{12}\,, \tag{3.5}$$

is a prime. Let $f(x) = x^2 - x + 1 \in \mathbb{Z}[x]$ and let r_1, r_2 be roots of $f(x) \equiv 0 \pmod{q}$. Note that r_1, r_2 exist if -3 is a quadratic residue modulo q. It follows from the law of quadratic reciprocity that $(\frac{-1}{q}) = 1$ if and only if $q \equiv 1 \pmod{3}$. Let p be a prime such that $p \equiv 2 \pmod{3}$, $p \equiv r_i \pmod{q}$, where $i = 1$ or $i = 2$. Because $q \equiv 1 \pmod{3}$ and $(3, q) = 1$, then by *The Chinese Remainder Theorem* we obtain

$$\begin{aligned} &p \equiv q + r_i \pmod{3q} \text{ for } r_i \equiv 1 \pmod{3}, \\ &p \equiv 2q + r_i \pmod{3q} \text{ for } r_i \equiv 0 \pmod{3}, \\ &p \equiv r_i \pmod{3q} \text{ for } r_i \equiv 2 \pmod{3}. \end{aligned} \tag{3.6}$$

It is easy to check that the algorithm in loops $1-5$ and $10-14$ generates primes q and p given respectively by (3.5) and (3.6) and in step 6 finds roots of $f(x) \pmod{q}$. A trivial verification shows that p and q are key parameters for the XTR cryptosystem.

We now estimate the complexity of the algorithm. Since $q \equiv 3 \pmod{4}$, roots of $f(x) \pmod{q}$ can be found using a single $\frac{q+1}{4}$th powering modulo q [9]. Then one can compute r_1, r_2 in $O(\log^3 N)$ bit operations. Now we check that $(3q, r_j^0) = 1$, $j = 0, 1, 2$, where $r_j^0 = q\big(2 - \big(r_i \pmod{3}\big)\big) + r_i$ and $r_i \equiv j \pmod{3}$. Since $q \equiv 1 \pmod{3}$, $r_j^0 \equiv 2 \pmod{3}$ and $(3, r_j^0) = 1$ for $j = 0, 1, 2$. Because $0 < r_i < q$, $(q, r_j^0) = 1$ for $j = 0, 1, 2$. By the above we conclude that $(3q, r_j^0) = 1$ for $j = 0, 1, 2$. Then the Theorem 3.7 implies that in loop $10-14$ one can find a prime p after repeating no more than $O(\log^2 N)$ steps of the algorithm.

We conclude from Theorems 3.7 and 3.6 that there exist such constants d_1 and N_1 that for every $N \geqslant N_1$ and arbitrary real $\lambda \geqslant 1$ we find with probability greater than or equal to $1 - e^{\lambda}$ a prime q and a prime p after repeating not more than $\big[d_1(\lambda + \log N)\log N\big]$ loops of the algorithm. Every step of the algorithm takes not more than $\mathcal{PT}$ bit operations. □

Corollary 3.9. *If $r_i \equiv 2 \pmod{3}$ is a prime for $i = 1$ or $i = 2$, then there exist such constants d_0 and N_0 that for every integer $N \geqslant N_0$ and an arbitrary real $\lambda \geqslant 1$ Algorithm 3.5 generates such primes p and q, key parameters for the XTR cryptosystem, that*

$$p = r_i, \quad N^{\frac{1}{2}} \ll p \ll N, \quad p \leqslant q \leqslant 2N,$$

with probability greater than or equal to $1 - e^{-\lambda}$ after repeating $[d_0 \lambda \log N] + 1$ loops of the algorithm. Every step of the algorithm takes no more than $\mathcal{PT}$ bit operations.

P r o o f. It is easy to check that if $r_i \equiv 2 \pmod{3}$ for $i = 1$ or $i = 2$ is a prime, then the algorithm returns $p = r_i$ and $q \mid p^2 - p + 1$. We conclude from Theorems 3.6 that there exist such constants d_0 and N_0 that for every $N \geqslant N_0$ and arbitrary real $\lambda \geqslant 1$ we find with probability greater than or equal to $1 - e^{\lambda}$ a prime q and a prime p after repeating not more than $[d_0 \lambda \log N] + 1$ loops of the algorithm. Every step of the algorithm takes no more than $\mathcal{PT}$ bit operations. $\square$

REFERENCES

[1] AGRAWAL, M.—KAYAL, K.—SAXENA, N.: *Primes is P*, Ann. of Math. **160** (2004), 781–793.

[2] BATEMAN, P. T.—HORN, R. A.: *A heuristic asymptotic formula concerning the distribution of prime numbers*, Math. Comp. **16** (1962), 119–132.

[3] BATEMAN, P. T.—HORN, R. A.: *Primes represented by irreducible polynomials in one variable*, Proc. Symposia in Pure Mathematics **8** (1965), 119–132.

[4] HEATH-BROWN, D. R.: *Almost-primes in arithmetic progression and short intervals*, Proc. London Proc. Cambridge Phil. Soc. **83** (1978), 357–375.

[5] HEATH-BROWN, D. R.: *Zero-free regions for Dirichlet L-functions and the least prime in an arithmetic progressions*, Proc. London Math. Soc. **64** (1992), 265–338.

[6] KANOLD, H. J.: *Elementare Betrachtungen zur Primzahltheorie*, Acta Math. **14** (1963), 147–151.

[7] LENSTRA, A. K.—VERHEUL, E. R.: *The XTR public key system*, in: Proceedings of CRYPTO '00, Lecture Notes in Comput. Sci., Vol. 1880, Springer-Verlag, Berlin, 2000, pp. 1–19.

[8] LINNIK, YU. V.: *On the last prime in an arithmetic progression. I. The basic theorem*, Mat. Sbornik, N. S. **15** (1944), 139–178.

[9] MENEZES, A. J.—VAN OORSCHOT, P. C.—VANSTONE, S. A.: *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1996.

[10] MILLER, G. L.: *Riemann's hypothesis and tests for primality*, J. Comput. Systems Sci. **13** (1976), 300–317.

[11] NARKIEWICZ, W.: *Classical Problems in Number Theory*, PWN, Warszawa, 1986.

[12] RABIN, M. O.: *Probabilistic algorithm for testing primality*, J. Number Theory **12** (1980), 128–138.

[13] SCHINZEL, A.—SIERPIŃSKI, W.: *Sur certaines hypothéses concernant les nombres premiers*, Acta Arith. **4** (1956), 185–208.

[14] TITCHMARSH, E. C.: *A. divisor problem*, Rend. Circ. Mat. Palermo **54** (1943), 87–105.

[15] WAGSTAFF, S. S., Jr.: *Greatest of the least primes in arithmetic progressions having a given modulus*, Math. Comp. **33** (1979), 1073–1080.

Received February 9, 2005

Cryptography Laboratory
Department of Mathematical
and Computer Sciences
University of Poznań
Umultowska 87
PL–61-614 Poznań
POLAND

E-mail: maciejg@amu.edu.pl