MATHEMATICS OF COMPUTATION
VOLUME 61, NUMBER 203
JULY 1993, PAGES 475-483

A REFINEMENT OF H. C. WILLIAMS' qth ROOT ALGORITHM

KENNETH S. WILLIAMS AND KENNETH HARDY

Dedicated to the memory of D. H. Lehmer

ABSTRACT. Let p and q be primes such that $p \equiv 1 \pmod{q}$. Let a be an integer such that $a^{(p-1)/q} \equiv 1 \pmod{p}$. In 1972, H. C. Williams gave an algorithm which determines a solution of the congruence $x^q \equiv a \pmod{p}$ in $O(q^3 \log p)$ steps, once an integer b has been found such that $(b^q - a)^{(p-1)/q} \not\equiv 0, 1 \pmod{p}$. A step is an arithmetic operation $(\bmod\ p)$ or an arithmetic operation on q-bit integers. We present a refinement of this algorithm which determines a solution in $O(q^4)+O(q^2 \log p)$ steps, once b has been determined. Thus the new algorithm is better when q is small compared with p.

1. INTRODUCTION

Let p and q be primes and let a be an integer not divisible by p. If $p \not\equiv 1 \pmod{q}$, the congruence

$$x^q \equiv a \pmod{p} \tag{1.1}$$

has one solution $x = a^u$, where u and v are integers such that $qu-(p-1)v = 1$. The integer u is easily found by applying the Euclidean algorithm to q and $p-1$. If $p \equiv 1 \pmod{q}$ and $a^{(p-1)/q} \not\equiv 1 \pmod{p}$, the congruence (1.1) has no solutions. If $p \equiv 1 \pmod{q}$ and $a^{(p-1)/q} \equiv 1 \pmod{p}$, (1.1) has q solutions. H. C. Williams [14] has given an algorithm for finding a solution x of (1.1) when q is odd. Briefly, his algorithm may be described as follows: first determine by trial an integer b such that $b^q - a$ is not a qth power residue of p; then use the formula

$$U_{j,m+n} \equiv \sum_{i=0}^{j} U_{i,n} U_{j-i,m} + (a-b^q) \sum_{i=1}^{q-1-j} U_{j+i,n} U_{q-i,m} \pmod{p}$$
$$(j = 0, 1, \dots, q-1;\ m = 1, 2, \dots;\ n = 1, 2, \dots)$$

recursively, starting with the initial values

$$U_{0,1} = b, \quad U_{1,1} = 1, \quad U_{j,1} = 0 \qquad (j = 2, \dots, q-1),$$

to compute $x = U_{0,(p^q-1)/((p-1)q)}$. Then x is a solution of (1.1). Once b has been determined, Williams' algorithm requires $O(q^3 \log p)$ steps to solve (1.1),

Received by the editor April 12, 1990 and, in revised form, May 15, 1990 and June 5, 1991.
1991 *Mathematics Subject Classification*. Primary 11Y16; Secondary 11A07, 11A15.
The first author's research was supported by Natural Sciences and Engineering Research Council of Canada Grant A-7233, and the second author's by Natural Sciences and Engineering Research Council of Canada Grant A-7823.

where by a step we mean an arithmetic operation in $GF(p)$ or an arithmetic operation on q-bit integers. All q solutions of the congruence (1.1) are given by

$$x(b^q - a)^{j(p-1)/q}, \qquad j = 0, 1, \dots, q-1.$$

In this paper we present a refinement of Williams' algorithm which determines a solution of (1.1) in $O(q^4) + O(q^2 \log p)$ steps, once the integer b has been found. Thus, our algorithm is better when q is small compared with p, roughly when $q = O_\varepsilon((\log p)^{1-\varepsilon})$, where $0 < \varepsilon < \frac{1}{2}$.

We remark that Williams' algorithm is a qth power version of an algorithm for computing square roots in $GF(p)$, which was published by Cipolla [5] in 1903 (see also [2], [9, pp. 132–134]). Shanks [12] has also given an algorithm for determining qth roots in $GF(p)$. His algorithm is an extension of an algorithm of Tonelli [13]. Adleman, Manders, and Miller [1] have shown, *assuming the extended Riemann hypothesis*, that there is a deterministic algorithm running in time $O(n \log^c(p+a))$ for some $c > 0$ such that on inputs a, p, n, where p is prime, it outputs the least positive integer x such that $x^n \equiv a \pmod p$ or "no" if no such x exists. It is an open problem to find a polynomial-time algorithm—polynomial in $\log q$ and $\log p$—for qth roots in $GF(p)$.

Algorithms for the more general problem of factoring polynomials over finite fields have been given by a number of authors, notably, Berlekamp [3], Moenck [10], Rabin [11], and Cantor and Zassenhaus [4] (see also [8, Chapter 4]). Cantor and Zassenhaus give a heuristic argument to suggest that the expected running time of their algorithm to factor a polynomial of degree n in $GF(p^m)$ is $O(n^3 + n^2 \log(p^m))$.

2. Idea of algorithm

Let p and q be primes with $q | p-1$. Let a be a nonzero element of $k = GF(p)$ which is the qth power of an element in k. We wish to determine a qth root of a. The algorithm constructs an extension field $K = k[\theta] \simeq GF(p^q)$ together with an element $\alpha \in K$ which, when raised to the power $(p^q - 1)/(p-1)$, gives a. It then follows that $\alpha^{(p^q-1)/(q(p-1))} \in k$ is the desired qth root of a. This strategy, in rather disguised form, is used by Williams [14]. The contribution of this paper is a way to compute the high power of α somewhat more quickly than the usual repeated squaring algorithm does. The idea is to write the exponent in base p and use automorphisms of K/k to get the effect of raising elements to p^eth powers.

3. The algorithm

Let p and q be primes satisfying $p \equiv 1 \pmod q$. Let $a \in k\backslash\{0\}$ be such that $a^{(p-1)/q} = 1$. We first show that there exists $b \in k$ with $(b^q - a)^{(p-1)/q} \neq 0, 1$. Clearly, we can identify k with the residues $\{1, 1-a, 1-2a, \dots, 1-(p-1)a\}$ modulo p. As k contains $(p-1)(q-1)/q \geq q-1 \geq 1$ elements which are not qth powers, we can let l be the smallest nonnegative integer such that $1 - la$ is not a qth power of an element of k. Clearly, we have $l \geq 1$ and $1 - (l-1)a = b^q$ for some $b \in k$. Then we have $b^q - a = 1 - la$, and so, as $1 - la$ is not a qth power, we have $(b^q - a)^{(p-1)/q} \neq 0, 1$. We set

$$c = (b^q - a)^{(p-1)/q}. \tag{3.1}$$

Clearly, c is a primitive qth root of unity in k. Since $b^q - a$ is not a qth power in k, we can adjoin a qth root θ of this quantity to k and obtain an extension field

$$K = k[\theta] = GF(p)[\theta] \simeq GF(p^q), \quad \text{where } \theta^q = b^q - a. \tag{3.2}$$

In K we have $\theta^p = (\theta^q)^{(p-1)/q}\theta = (b^q - a)^{(p-1)/q}\theta = c\theta$, so that

$$\theta^{p^n} = c^n\theta, \qquad n = 0, 1, 2, \dots . \tag{3.3}$$

Now define $x \in K$ by

$$x = (b - \theta)^{(p^q-1)/((p-1)q)}. \tag{3.4}$$

As $(p^q - 1)/(p - 1) = 1 + p + p^2 + \cdots + p^{q-1}$, we have

$$x^q = \prod_{j=0}^{q-1} (b - \theta)^{p^j}. \tag{3.5}$$

Next we observe that $(b - \theta)^p = b^p - \theta^p = b - c\theta$, so that

$$(b - \theta)^{p^j} = b - c^j\theta, \qquad j = 0, 1, 2, \dots . \tag{3.6}$$

As c is a primitive qth root of unity in k, we have

$$\prod_{j=0}^{q-1} (b - c^j\theta) = b^q - \theta^q = a, \tag{3.7}$$

so that by (3.5), (3.6), and (3.7), we see that $x^q = a$. Since the equation $y^q = c$ has at most q solutions in the field K, and since it has exactly q solutions in the subfield k, every solution must belong to k. Thus, in particular, we have $x \in k$. We have thus shown that $x = (b - \theta)^{((p^q-1)/((p-1)q))}$ is a qth root of a in k. We remark that H. C. Williams' algorithm is equivalent to computing $\frac{1}{q}\mathrm{tr}_{K/k}((b - \theta)^{(p^q-1)/(p-1)q})$, which is also a qth root of a. Note also that $N_{K/k}(b - \theta) = a$.

In order to compute x, we write it in the form

$$x = E_1^{(p-1)/q} E_2, \tag{3.8}$$

where

$$E_1 = (b - \theta)^{(p-1)^{q-2}}, \qquad E_2 = (b - \theta)^{(p^q-1)/q(p-1) - (p-1)^{q-1}/q}. \tag{3.9}$$

First we consider E_1. Applying the binomial theorem to $(p - 1)^{q-2}$, and appealing to (3.6), we obtain

$$E_1 = \prod_{i=0}^{q-2} (b - c^i\theta)^{(-1)^{q-i}\binom{q-2}{i}}, \tag{3.10}$$

say

$$E_1 = \sum_{i=0}^{q-1} a_i\theta^i, \tag{3.11}$$

where $a_i \in k$, $i = 0, 1, \ldots, q-1$. Now define $a_i(j) \in k$ for $i = 0, 1, \ldots, q-1$ and $j = 1, 2, 3, \ldots$ by

$$\sum_{i=0}^{q-1} a_i(j)\theta^i = \left(\sum_{i=0}^{q-1} a_i\theta^i\right)^j, \tag{3.12}$$

so that

$$a_i(1) = a_i, \qquad i = 0, 1, \ldots, q-1, \tag{3.13}$$

and

$$E_1^{(p-1)/q} = \sum_{i=0}^{q-1} a_i((p-1)/q)\theta^i. \tag{3.14}$$

Next we consider E_2. Again, by the binomial theorem and (3.6), we obtain

$$E_2 = \prod_{i=1}^{q-1} (b - c^{q-i-1}\theta)^{(1-(-1)^i\binom{q-1}{i})/q}. \tag{3.15}$$

It is easily proved by induction on i that the exponent $(1-(-1)^i\binom{q-1}{i})/q$ is an integer. Thus we have

$$E_2 = \sum_{i=0}^{q-1} b_i\theta^i, \tag{3.16}$$

where $b_i \in k$, $i = 0, 1, \ldots, q-1$. From (3.8), (3.14), and (3.16), we deduce

$$x = \left(\sum_{i=0}^{q-1} a_i((p-1)/q)\theta^i\right)\left(\sum_{j=0}^{q-1} b_j\theta^j\right),$$

that is

$$x = a_0((p-1)/q)b_0 + (b^q - a)\sum_{i=1}^{q-1} a_i((p-1)/q)b_{q-i}. \tag{3.17}$$

Formula (3.17) is the expression we use to calculate x. We can now give the algorithm.

Algorithm to determine all solutions x of the congruence $x^q \equiv a \pmod p$.

Input. p, q primes satisfying $p \equiv 1 \pmod q$. a an integer not divisible by p.

Step 1. Compute $a^{(p-1)/q}$ in $k = GF(p)$. If $a^{(p-1)/q} \neq 1$, then $x^q = a$ has no solutions in k and the algorithm terminates. Otherwise, $x^q = a$ has q solutions in k and the algorithm continues with Step 2.

Step 2. Try $b = 1, 2, 3 \ldots$ until the first integer b is found such that $(b^q - a)^{(p-1)/q} \neq 0, 1$, and set $c = (b^q - a)^{(p-1)/q}$.

Step 3. In $K = k[\theta] = \{c_0 + c_1\theta + \cdots + c_{q-1}\theta^{q-1} | c_0, c_1, \ldots, c_{q-1} \in k\}$, where $\theta^q = b^q - a$, compute the quantities $X_i = (b - c^i\theta)^{(-1)^{q-i}\binom{q-1}{i}}$ for $i = 0, 1, \ldots, q-2$ and $Y_i = (b - c^{q-i-1}\theta)^{(1-(-1)^i\binom{q-1}{i})/q}$ for $i = 1, \ldots, q-1$. Then compute the products $E_1 = \prod_{i=0}^{q-2} X_i = \sum_{i=0}^{q-1} a_i\theta^i$ and $E_2 = \prod_{i=1}^{q-1} Y_i = \sum_{i=0}^{q-1} b_i\theta^i$ to obtain $a_0, a_1, \ldots, a_{q-1}, b_0, b_1, \ldots, b_{q-1} \in k$.

Step 4. Use the recurrence relation in k,

$$a_i(m+n) = \sum_{j=0}^{i} a_j(m)a_{i-j}(n) + (b^q - a)\sum_{j=i+1}^{q-1} a_j(m)a_{q+i-j}(n) \qquad (m, n = 1, 2, \dots), \tag{3.18}$$

subject to the initial conditions

$$a_i(1) = a_i \qquad (i = 0, 1, \dots, q-1), \tag{3.19}$$

to calculate $a_i((p-1)/q)$ $(i = 0, 1, \dots, q-1)$.

Output. A solution x of the congruence $x^q \equiv a \pmod{p}$ is given by

$$x = a_0((p-1)/q)b_0 + (b^q - a)\sum_{i=1}^{q-1} a_i((p-1)/q)b_{q-i}. \tag{3.20}$$

All solutions are given by $x_j = c^j x$, $j = 0, 1, \dots, q-1$.

We conclude this section by determining the running time of the algorithm. Recall that a step is an arithmetic operation in $k = GF(p)$ or an arithmetic operation on q-bit integers. Note that arithmetic operations in $K = GF(p^q)$ take $O(q^2)$ steps.

Step 1. The calculation of $a^{(p-1)/q}$ can be carried out in $O(\log p)$ steps in k by the repeated squaring technique.

Step 2. Let N denote the number of $(x, y) \in k \times k$ with $x^q - y^q = a$ and B the number of values of $b \in k$ for which $(b^q - a)^{(p-1)/q} = 0$ or 1. Then we have

$$N = \sum_{\substack{x\in k \\ x^q - y^q = a}} \sum_{y \in k} 1 = \sum_{\substack{x \in k \\ x^q = a}} 1 + \sum_{x \in k} \sum_{\substack{y \in k \\ y \neq 0 \\ y^q = x^q - a}} 1 = q + q \sum_{\substack{x \in k \\ (x^q - a)^{(p-1)/q} = 1}} 1 = q + q(B - q).$$

From the work of Davenport and Hasse [6, p. 174] we have

$$|N - p| \le q - 1 + ((q-1)^2 - (q-1))\sqrt{p},$$

so that

$$|qB - q(q-1) - p| \le q - 1 + (q-1)(q-2)\sqrt{p}.$$

Hence, we have

$$\begin{aligned} B &\le \frac{p}{q} + \frac{(q^2-1)}{q} + \frac{(q-1)(q-2)}{q}\sqrt{p} \\ &\le \frac{p}{q} + \frac{(2q^2 - 3q + 1)}{q}\sqrt{p} \le \frac{p}{q} + 2q\sqrt{p} \end{aligned}$$

and

$$B \ge \frac{p}{q} + \frac{(q-1)^2}{q} - \frac{(q-1)(q-2)}{q}\sqrt{p} \ge \frac{p}{q} - q\sqrt{p},$$

so that

$$\left|\frac{B}{p} - \frac{1}{q}\right| \le \frac{2q}{\sqrt{p}}.$$

Thus, for q small compared with p, say for example $q \le p^{1/4}$, a random value of b does not satisfy $(b^q - a)^{(p-1)/q} \neq 0, 1$ with probability

$$\frac{B}{p} = \frac{1}{q} + O\left(\frac{q}{\sqrt{p}}\right) = \frac{1}{q} + O(p^{-1/4}).$$

Thus finding an appropriate value of b is usually quite fast in practice.

Step 3. First we observe that all of the values of b^i $(i = 0, 1, \ldots, q-1)$ and c^i $(i = 0, 1, \ldots, (q-1)^2)$ can be computed in $O(q^2)$ arithmetic operations in k.

Next we remark that as

$$\binom{q-2}{i} \le 2^{q-2} < 2^q < 10^q$$

for $i = 0, 1, \ldots, q-2$, each entry in the first $q-2$ rows of Pascal's triangle can be represented as a q-bit integer, and so $O(q^2)$ additions of q-bit integers are required to compute all the binomial coefficients $\binom{q-2}{i}$ $(i = 0, 1, \ldots, q-2)$ from Pascal's triangle.

Knowing the values of c^i $(i = 0, 1, \ldots, q-2)$ and $\binom{q-2}{i}$ $(i = 0, 1, \ldots, q-2)$, we can, when $q-i$ is even, compute each quantity $(b - c^i\theta)^{(-1)^{q-i}\binom{q-2}{i}} = (b - c^i\theta)^{\binom{q-2}{i}}$ by repeated squarings in K in $O(q^2 \log\binom{q-2}{i})$ steps. Knowing the values of b^i $(i = 0, 1, \ldots, q-1)$, $(c^i)^j$ $(i = 0, 1, \ldots, q-1;\ j = 0, 1, \ldots, q-1)$ and $\binom{q-2}{i}$ $(i = 0, 1, \ldots, q-2)$, as

$$(3.21) \qquad (b - c^i\theta)^{-1} = a^{p-2}(b^{q-1} + b^{q-2}c^i\theta + \cdots + (c^i)^{q-1}\theta^{q-1}),$$

we can, when $q-i$ is odd, compute each quantity

$$\begin{aligned}(b - c^i\theta)^{(-1)^{q-i}\binom{q-2}{i}} &= (b - c^i\theta)^{-\binom{q-2}{i}} \\ &= (a^{p-2}(b^{q-1} + b^{q-2}c^i\theta + \cdots + (c^i)^{q-1}\theta^{q-1}))^{\binom{q-2}{i}}\end{aligned}$$

by repeated squarings in K in

$$O(\log p) + O(q) + O\left(q^2 \log\binom{q-2}{i}\right)$$

steps. Hence, all of

$$X_i = (b - c^i\theta)^{(-1)^{q-2-i}\binom{q-2}{i}} \qquad (i = 0, 1, \ldots, q-2)$$

can be computed in

$$O(q^2) + \sum_{i=0}^{q-2}\left(O(\log p) + O(q) + O\left(q^2 \log\binom{q-2}{i}\right)\right) = O(q\log p) + O(q^4)$$

steps, as

$$\sum_{i=0}^{n} \log\binom{n}{i} \sim \frac{1}{2}n^2, \quad \text{as } n \to \infty,$$

see [7]. Multiplying the X_i together in K to obtain $E_1 = \prod_{i=0}^{q-2} X_i = \sum_{i=0}^{q-1} a_i\theta^i$ takes a further $O(q)$ multiplications in K, that is, $O(q^3)$ steps. Hence, $a_0, a_1, \ldots, a_{q-1}$ can be computed in $O(q\log p) + O(q^4)$ steps. A similar calculation shows that $b_0, b_1, \ldots, b_{q-1}$ can also be computed in $O(q\log p) + O(q^4)$ steps.

Step 4. The quantities $a_i((p-1)/q)$ $(i = 0, 1, \ldots, q-1)$ can be computed from the values of the a_i $(i = 0, 1, \ldots, q-1)$ using (3.18) in $O(q^2 \log p)$ steps, since each use of the recurrence relation (3.18) requires $O(q)$ operations and each of the q recurrence relations must be applied $O(\log(p-1)/q)$ times in the repeated doubling technique.

The calculation of the solution x of (1.1) from the values of the $a_i((p-1)/q)$ $(i = 0, 1, \ldots, q-1)$ and b_i $(i = 0, 1, \ldots, q-1)$ using (3.20) takes $O(q)$ steps, and the calculation of the other solutions xc^j $(j = 1, 2, \ldots, q-1)$ can be done in $O(q)$ steps. Hence the algorithm determines all the solutions of (1.1) in $O(q^4) + O(q^2 \log p)$ steps, once a suitable b has been determined in Step 2.

We remark that this algorithm (suitably modified) can be used to compute qth roots in $GF(p^n)$, when q divides $p^n - 1$.

4. Example

Following the suggestion of the referee, we present a small example to illustrate our algorithm, which the interested reader can easily check by hand. The algorithm is easily programmed to solve (1.1) for large values of p and values of q small compared with p.

We determine all the solutions x of the congruence

$$x^3 \equiv 2 \pmod{31}, \tag{4.1}$$

using our refinement to the algorithm of H. C. Williams. Here, $p = 31$, $q = 3$, $a = 2$, $(p-1)/q = 10$, $(p-1)^{q-2} = 30$ and $(p^q-1)/q(p-1)-(p-1)^{q-1}/q = 31$. As

$$a^{(p-1)/q} = 2^{10} \equiv 32^2 \equiv 1^2 \equiv 1 \pmod{p},$$

the congruence (4.1) is solvable. We can take $b = 2$, $c = 25$, as

$$(b^q - a)^{(p-1)/q} = (2^3 - 2)^{10} = 6^{10} = 36^5 \equiv 5^5 \equiv 3125 \equiv 25 \pmod{p}.$$

Also, θ is a root of $\theta^q = b^q - a$, that is, $\theta^3 = 6$. We perform calculations in $k = GF(31)$ and $K = GF(31)[\theta] \simeq GF(31^3)$.

Appealing to (3.9), (3.10), and (3.21), we have

$$E_1 = (2-\theta)^{30} = (2-\theta)^{-1}(2-25\theta) = (2+\theta+16\theta^2)(2+6\theta) = 22 + 14\theta + 7\theta^2,$$

so that $a_0 = 22$, $a_1 = 14$, $a_2 = 7$. Making use of the recurrence relations

$$\begin{cases} a_i(m+n) = \sum_{j=0}^{i} a_j(m)a_{i-j}(n) + 6\sum_{j=i+1}^{2} a_j(m)a_{3+i-j}(n), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad m, n = 1, 2, \ldots, \\ a_i(1) = a_i, \qquad i = 0, 1, 2, 3, \end{cases}$$

we obtain the values in Table 1 (next page).

Next, from (3.9) and (3.15), we have $E_2 = 2 - 25\theta$, so that $b_0 = 2$, $b_1 = 6$, $b_2 = 0$. Finally, appealing to (3.20), we obtain

$$x = a_0(10)b_0 + 6(a_1(10)b_2 + a_2(10)b_1) = 9 \times 2 + 6 \times 19 \times 6 = 20.$$

TABLE 1

j	$a_0(j)$	$a_1(j)$	$a_2(j)$
1	22	14	7
2	17	11	8
4	12	14	21
8	14	6	18
10	9	4	19

We note that $x = 20$ is indeed a solution of (4.1), as $20^3 \equiv (-11)^3 \equiv (-121)11 \equiv 3 \times 11 = 33 \equiv 2 \pmod{31}$. All solutions of (4.1) are given by $x \equiv 20 \cdot 25^j \pmod{31}$, $j = 0, 1, 2$, that is, $x \equiv 20, 4, 7 \pmod{31}$.

ACKNOWLEDGMENTS

The authors would like to thank an unknown referee for valuable comments which corrected and greatly improved the first version of this paper.

The authors would also like to acknowledge the help of Dr. B. K. Spearman (Okanagan University College) and Mr. N. Buck (College of New Caledonia) in connection with the preparation of this paper.

BIBLIOGRAPHY

1. L. Adleman, K. Manders, and G. Miller, *On taking roots in finite fields*, Proc. 18th Ann. Sympos. Found. Comp. Sci. (Providence, RI, 1977), IEEE Comput. Sci., Long Beach, CA, 1977, pp. 175–178.

2. W. S. Anglin, $x^2 \equiv R \pmod{p}$, Preprint, McGill University, 1987.

3. E. R. Berlekamp, *Factoring polynomials over large finite fields*, Math. Comp. **24** (1970), 713–735.

4. D. G. Cantor and H. J. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Math. Comp. **36** (1981), 587–592.

5. M. Cipolla, *Un metodo per la risoluzione della congruenza di secondo grado*, Rend. Accad. Sci. Fis. Mat. Napoli (3) **9** (1903), 154–163.

6. H. Davenport and H. Hasse, *Die Nullstellen der Kongruenzzetafunktionen in gewissen zyklischen Fällen*, J. Reine Angew. Math. **172** (1934), 151–182.

7. H. W. Gould, *Sums of logarithms of binomial coefficients*, Amer. Math. Monthly **71** (1964), 55–58.

8. D. E. Knuth, *The art of computer programming*, Vol. 2, *Seminumerical algorithms*, Addison-Wesley, Reading, MA, 1969.

9. D. H. Lehmer, *Computer technology applied to the theory of numbers*, Studies in Number Theory (W. J. LeVeque, ed.), MAA Studies in Math., vol. 6, Math. Assoc. Amer., Washington, DC, 1969, pp. 117–151.

10. R. T. Moenck, *On the efficiency of algorithms for polynomial factoring*, Math. Comp. **31** (1977), 235–250.

11. M. O. Rabin, *Probabilistic algorithms in finite fields*, SIAM J. Comput. **9** (1980), 273–280.

12. D. Shanks, *Five number-theoretic algorithms*, Proc. Second Manitoba Conf. on Numerical Mathematics, University of Manitoba, Winnipeg, Canada, 1972, pp. 51–70.

13. A. Tonelli, *Bemerkung über die Auflösung quadratischer Congruenzen*, Gött. Nachr. (1891), 344–346.

14. H. C. Williams, *Some algorithms for solving* $x^q \equiv N \pmod{p}$, Proc. 3rd Southeastern Conf. on Combinatorics, Graph Theory, and Computing (Florida Atlantic University, 1972), 451–462.

Department of Mathematics and Statistics, Carleton University, Ottawa, Ontario, Canada K1S 5B6

E-mail address: mathstat@carleton.ca