Square Roots Modulo p

Gonzalo Tornaría [⋆]

Department of Mathematics,
University of Texas at Austin,
Austin, Texas 78712, USA,
`tornaria@math.utexas.edu`

Abstract. The algorithm of Tonelli and Shanks for computing square roots modulo a prime number is the most used, and probably the fastest among the known algorithms when averaged over all prime numbers. However, for some particular prime numbers, there are other algorithms which are considerably faster.
In this paper we compare the algorithm of Tonelli and Shanks with an algorithm based in quadratic field extensions due to Cipolla, and give an explicit condition on a prime number to decide which algorithm is faster. Finally, we show that there exists an infinite sequence of prime numbers for which the algorithm of Tonelli and Shanks is asymptotically worse.

1 Introduction

When p is an odd prime, we denote by $\mathbb{F}_p$ the finite field of p elements, and by $\mathbb{F}_p^\times$ its multiplicative group. We will consider two algorithms for solving the following:

Problem. Let p be an odd prime, and $a \in \mathbb{F}_p$ a quadratic residue. Find $x \in \mathbb{F}_p$ such that $x^2 = a$.

From now on p will be a fixed prime number. Write $p = 2^e q + 1$ with q odd; this determines e and q. Let n be the number of binary digits of p, and let k be the number of ones in the binary representation of p. We denote by $\mathbf{G}_p$ the Sylow 2-subgroup of $\mathbb{F}_p^\times$, which is cyclic of order 2^e.

2 The Algorithm of Tonelli and Shanks

The algorithm of Tonelli [9], is based in this observation: it's easy to reduce the problem to the case $a \in \mathbf{G}_p$, because $[\mathbb{F}_p^\times : \mathbf{G}_p]$ is odd. Then one can use the Legendre symbol to find a generator of $\mathbf{G}_p$, and compute the square root of a by means of the discrete logarithm of a with respect to that generator, which is fast because most of the time the group $\mathbf{G}_p$ is small. Moreover, there is a binary algorithm for computing discrete logarithms in a cyclic group of order 2^e; namely, compute the discrete logarithm bit by bit.

[⋆] This work was partially supported by a scholarship of PEDECIBA Matemática

The original method of Tonelli required about $e^2/2$ operations for computing the discrete logarithm. It was improved by Shanks [8], who rearranged the algorithm in a clever way such that the operations done for computing a 0 bit include the operations needed for computing the next bit. Thus, while the number of operations in the worst case is the same, the number is roughly halved in average.

Indeed one has

Proposition 2.1 (Lindhurst [5, Lemma 1]). *Averaged over all quadratic residue and non-residue inputs, and ignoring the initialization stage, the algorithm of Tonelli and Shanks requires $\frac{1}{4}(e^2+7e-12)+\frac{1}{2^{e-1}}$ modular multiplications.* $\square$

Remark. In this result the average is taken over a uniformly distributed input $a \in \mathbb{F}_p^\times$; if the distribution of a is a concern, one can compute instead $\frac{\sqrt{ab^2}}{b}$, where b is a uniformly distributed random element of $\mathbb{F}_p^\times$.

Of course, the density of the prime numbers $p = 2^e q + 1$ for a fixed e is, by Dirichlet's theorem, 2^{-e}, and one can conclude that:

Corollary 2.2 (Lindhurst [5, Theorem 2]). *Averaged over all prime numbers, quadratic residues and non-residues, the algorithm of Tonelli and Shanks requires $8/3$ multiplications after the initialization stage.* $\square$

Remark. The average over all prime numbers is, as usual, the limit for $N \to \infty$ of the (uniform) average over all primes $\leq N$.

The initialization stage has two steps:

- Find a generator of $\mathbf{G}_p$. For this we take $t \in \mathbb{F}_p^\times$ at random and compute the Legendre symbol of t until we get a quadratic non-residue. Then $z = t^q$ is a generator of $\mathbf{G}_p$. The probability of getting a quadratic non-residue is $1/2$ for each try; the expected number of Legendre symbol computations is therefore 2. Also, q has $n-e$ binary digits, of which $k-1$ are ones, so computing t^q will require $n+k-e-3$ multiplications.
- Compute $a^{(q+1)/2}$ and a^q. The binary expression of $(q-1)/2$ has $n-e-1$ digits, of which $k-2$ are ones. Using a binary powering algorithm, one can compute $a^{(q-1)/2}$ with $n+k-e-5$ multiplications, and we need one more for computing $a^{(q+1)/2}$ and another one to compute $a^q = a^{(q-1)/2}a^{(q+1)/2}$.

Corollary 2.3. *Averaged over all quadratic residue and non-residue inputs, the number of operations required by the algorithm of Tonelli and Shanks is*

$$2n + 2k + \frac{e(e-1)}{4} + \frac{1}{2^{e-1}} - 9 \tag{1}$$

multiplications, and 2 expected (with respect to choosing t, which is independent of the input) computations of the Legendre symbol. $\square$

These results support our assertion that the algorithm of Tonelli and Shanks is very good when the modulus is fairly random, but they also show that there could be room for improvements when e is large with respect to n, provided that such prime numbers exist.

3 The Algorithm of Cipolla

An alternative to using discrete logarithms is the algorithm of Cipolla [3]. Let $a \in \mathbb{F}_p^\times$, and assume that we know $t \in \mathbb{F}_p$ such that $t^2 - a$ is a quadratic non-residue. Then $X^2 - (t^2 - a)$ is irreducible over $\mathbb{F}_p$, and $\mathbb{F}_p[\alpha]$, with $\alpha^2 = t^2 - a$, is a finite field of p^2 elements, a quadratic extension of $\mathbb{F}_p$.

It's enough to compute the square root of a in $\mathbb{F}_p[\alpha]$. If a is a quadratic residue, its two square roots will be in $\mathbb{F}_p$; otherwise we will get something not in $\mathbb{F}_p$, and we will be able to conclude that a is a quadratic non-residue.

In $\mathbb{F}_p[\alpha]$ we have

$$(t+\alpha)^{p+1} = (t+\alpha)(t+\alpha)^p = (t+\alpha)(t-\alpha) = t^2 - \alpha^2 = a \ , \qquad (2)$$

where the second equality follows because the Frobenius automorphism carries $t+\alpha$ to its conjugate, $t-\alpha$. Therefore, if we compute $x = (t+\alpha)^{(p+1)/2}$, we have $x^2 = a$.

To find t, we just take $t \in \mathbb{F}_p^\times$ at random and compute the Legendre symbol of $t^2 - a$ until we get one such that $t^2 - a$ is a quadratic non-residue or 0 (if we are so lucky, t itself is a square root).

Lemma 3.1. *Let $a \in \mathbb{F}_p^\times$. The number of $t \in \mathbb{F}_p$ such that $t^2 - a$ is a quadratic non-residue is $(p-1)/2$ if a is a quadratic residue and $(p+1)/2$ if a is a quadratic non-residue.*

Proof. Notice that the set of t such that $t^2 - a$ is a quadratic residue is exactly the same as the set of different t which appear among the pairs (s,t) such that $s^2 = t^2 - a$. This equation is the same as $(t-s)(t+s) = a$, and so it clearly has $p-1$ solutions.

Now, for each solution (s,t) we get a different solution $(-s,t)$ with the same t, unless $s = 0$. We have two cases to consider:

- If a is a quadratic residue, then there are two different solutions with $s = 0$, and so the number of different t which appear in the set of solutions is $(p-3)/2 + 2 = (p+1)/2$.
- If a is a quadratic non-residue, then there are no solutions with $s = 0$, and the number of different t is $(p-1)/2$. $\square$

This lemma shows that in practice is very easy to find such a t, the probability for each try being slightly more than $1/2$; the expected number of tries is therefore less than 2, and we only need one multiplication, one sum, and one computation of the Legendre symbol for each try.

After finding t, we only have to compute a $(p+1)/2$ power of $(t+\alpha)$ in $\mathbb{F}_p[\alpha]$. For this we can use a binary powering algorithm, using the following formulas for multiplication in $\mathbb{F}_p[\alpha]$:

- $(u + v\alpha)^2 = (u^2 + v^2 r) + ((u+v)^2 - u^2 - v^2)\alpha$, where $r = t^2 - a$ is known in advance, which needs 4 multiplications and 4 sums;

– $(u + v\alpha)^2(t + \alpha) = (td^2 - b(u + d)) + (d^2 - bv)\alpha$, where $d = (u + vt)$ and $b = av$, which needs 6 multiplications and 4 sums.

We can assume that $p \equiv 1 \pmod 4$, as Shanks' algorithm is obviously better otherwise. In that case, the binary expression of $(p + 1)/2$ has $n - 1$ digits, of which k are ones. For computing a $(p+1)/2$ power of $(t + \alpha)$ we therefore need to use $n - k - 1$ times the first formula, and $k - 1$ times the second formula.

Proposition 3.2. *For any quadratic residue or non-residue input, the expected number of operations required for the algorithm of Cipolla is $4n + 2k - 4$ multiplications, $4n - 2$ sums, and 2 computations of the Legendre symbol.*

Combining Corollary 2.3 and Proposition 3.2 we obtain

Theorem 3.3. *Given a prime number p, let n be the number of binary digits of p, and let 2^e be the maximum power of 2 which divides $p - 1$. With respect to the expected number of operations, and averaged over all quadratic residue and non-residue inputs, the algorithm of Cipolla (neglecting the sums) is better than the algorithm of Tonelli and Shanks if and only if $e(e - 1) > 8n + 20$.* □

4 The Existence of Primes

For each $i = 1, 2, \ldots$, we define p_i to be the least prime number such that $p_i \equiv 2^i + 1 \pmod{2^{i+1}}$. Let n_i be the number of binary digits of p_i, and let 2^{e_i} be the maximum power of 2 which divides $p_i - 1$. From the definition is clear that $e_i = i$, and $n_i > i$. We now give an upper bound for n_i.

Lemma 4.1. *There exists absolute constants L, C such that $e_iL + C > n_i$.*

Proof. A theorem of Linnik [6, 7] states that if $(a, m) = 1$ then the least prime number congruent to a modulo m is less than C_0m^L for some absolute constants C_0, L. Applying this to p_i we get

$$p_i < C_0 2^{(e_i+1)L} \quad . \tag{3}$$

Taking base 2 logarithms, we conclude that $n_i < e_iL + C$. □

Remark. The best known unconditional value for L is 11/2, due to Heath-Brown [4]. Assuming the Generalized Riemann Hypothesis, one can use $L = 2+\epsilon$ for arbitrary $\epsilon > 0$ [1, 4]. In the case in hand, where the modulus are all powers of two, there may be even stronger results. For example, in [1] the authors prove that one can use $L = 8/3 + \epsilon$ provided the modulus are restricted to powers of a fixed *odd* prime.

In the following theorem, $\mathsf{TS}(p)$ and $\mathsf{Cip}(p)$ are the expected number of operations required for the prime p by the algorithms of Tonelli and Shanks and by the algorithm of Cipolla respectively, averaged over all quadratic residue and non-residue inputs.

Theorem 4.2.

$$\limsup_{p\ prime} \frac{\mathsf{TS}(p)}{\mathsf{Cip}(p)} = \infty \ . \tag{4}$$

Proof. From Corollary 2.3 and Proposition 3.2 we know that

$$\mathsf{TS}(p_i) > \frac{e_i(e_i-1)}{4} > \frac{(n_i-C)(n_i-C-L)}{4L^2} = \Omega(n_i^2) \tag{5}$$

and that $\mathsf{Cip}(p_i) = O(n_i)$ (even counting the sums and the Legendre symbol computations). Therefore $\frac{\mathsf{TS}(p_i)}{\mathsf{Cip}(p_i)} = \Omega(n_i)$, and the theorem follows. □

5 Last Remarks

I thank the referee for pointing me to the work of Bernstein [2]. In this work, Bernstein improves the algorithm of Tonelli and Shanks. He computes discrete logarithms several bits at a time by means of some auxiliar precomputations. This is especially appealing if one needs to compute several square roots modulo the same prime number, but the improvement is still good for the casual use, if the number of bits computed at a time, and with it the amount of precomputation, is choosen apropriately.

Taking into account the precomputations for this new algorithm, Theorem 4.2 is still valid, but Theorem 3.3 would have to be changed; according to [2] the new algorithm is better than the algorithm of Cipolla when $e^2 = O(n(\lg n)^2)$.

References

1. Barban, M.B., Linnik, Y.V., Tshudakov, N.G.: On prime numbers in an arithmetic progression with a prime-power difference. Acta Arith. **9** (1964) 375–390
2. Bernstein, D.J.: Faster square roots in annoying finite fields, draft. Available from `http://cr.yp.to/papers.html` (2001)
3. Cipolla, M.: Un metodo per la risoluzione della congruenza di secondo grado. Rend. Accad. Sci. Fis. Mat. Napoli **9** (1903) 154–163
4. Heath-Brown, D.R.: Zero-free regions for Dirichlet L-functions, and the least prime in an arithmetic progression. Proc. London Math. Soc. (3) **64** (1992) 265–338
5. Lindhurst, S.: An analysis of Shanks's algorithm for computing square roots in finite fields. In: Number theory (Ottawa, ON, 1996). Amer. Math. Soc., Providence, RI (1999) 231–242
6. Linnik, U.V.: On the least prime in an arithmetic progression. I. The basic theorem. Rec. Math. [Mat. Sbornik] N.S. **15(57)** (1944) 139–178
7. Linnik, U.V.: On the least prime in an arithmetic progression. II. The Deuring-Heilbronn phenomenon. Rec. Math. [Mat. Sbornik] N.S. **15(57)** (1944) 347–368
8. Shanks, D.: Five number-theoretic algorithms. In: Proceedings of the Second Manitoba Conference on Numerical Mathematics (Univ. Manitoba, Winnipeg, Man., 1972). Utilitas Math., Winnipeg, Man. (1973) 51–70. Congressus Numerantium, No. VII
9. Tonelli, A.: Bemerkung über die Auflösung quadratischer Congruenzen. Göttinger Nachrichten (1891) 344–346