## 3.1 Quadratic reciprocity

Recall that for each odd prime $p$ the Legendre symbol $(\frac{a}{p})$ is defined as

$$\left(\frac{a}{p}\right)=\begin{cases}1 & \text{if } a \text{ is a nonzero quadratic residue modulo } p,\\ 0 & \text{if } a \text{ is zero modulo } p,\\ -1 & \text{otherwise.}\end{cases}$$

The Legendre symbol is multiplicative, $(\frac{a}{p})(\frac{b}{p})=(\frac{ab}{p})$, and it can be computed using Euler's criterion:

$$\left(\frac{a}{p}\right)\equiv a^{\frac{p-1}{2}} \bmod p.$$

Both statements follow from the fact that the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^\times$ is cyclic of order $p-1$ with $-1$ as the unique element of order 2 (the case $a=0$ is clear). We also have the well known law of quadratic reciprocity.

**Theorem 3.1** (Gauss)**.** *For all odd primes $p$ and $q$ we have* $(\frac{p}{q})(\frac{q}{p})=(-1)^{(\frac{p-1}{2})(\frac{q-1}{2})}$.

I expect you have all seen proofs of this theorem, but I recently came across the following proof due to Rousseau [4], which Math Overflow overwhelmingly voted as the "best" proof quadratic reciprocity. The proof is quite short, so I thought I would share it with you.

*Proof.* Let $s=(p-1)/2$, $t=(q-1)/2$ and $u=(pq-1)/2$. Consider the three subsets of $(\mathbb{Z}/pq\mathbb{Z})^\times$ defined by

$$A=\{x : x \bmod p\in[1,s]\},\quad B=\{x : x \bmod q\in[1,t]\},\quad C=\{x : x \bmod pq\in[1,u]\}.$$

These subsets each contain exactly half of the $(p-1)(q-1)=4st$ elements of $(\mathbb{Z}/pq\mathbb{Z})^\times$ and thus have size $2st$. Furthermore, for all $x\in(\mathbb{Z}/pq\mathbb{Z})^\times$ each subset contains exactly one of $x$ or $-x$. It follows that the products $a,b,c$ over the sets $A,B,C$ differ only in sign, so their ratios are all $\pm1$. The intersection of $A$ and $B$ has size $st$, hence there are $2st-st=st$ sign differences between the elements of $A$ and $B$, and therefore $a/b\equiv(-1)^{st} \bmod pq$. To complete the proof, we just need to show that $a/b\equiv(\frac{p}{q})(\frac{q}{p}) \bmod pq$, since two numbers that are both equal to $\pm1$ and congruent mod $pq>2$ must be equal over $\mathbb{Z}$.

Considering the product $a$ modulo $q$, it is clear that $a\equiv(q-1)!^s \bmod q$, since modulo $q$ we are just multiplying $s$ copies of the integers from 1 to $q-1$. To compute $c$ modulo $q$ we first compute the product of the integers in $[1,u]$ that are not divisible by $q$, which is $(q-1)!^s t!$, and then divide by the product of the integers in $[1,u]$ that are multiples of $p$, since these do not lie in $(\mathbb{Z}/pq\mathbb{Z})^\times$, which is $p\times2p\times\cdots tp=p^t t!$. Thus $c\equiv(q-1)!^s/p^t \bmod q$, and we have $a/c\equiv p^t\equiv\pm1 \bmod q$. But we know that $a/c\equiv\pm1 \bmod pq$, so this congruence also holds mod $pq$. By Euler's criterion, we have $a/c\equiv(\frac{p}{q}) \bmod pq$. Similarly, $b/c\equiv(\frac{q}{p}) \bmod pq$, and since $b/c\equiv\pm1 \bmod pq$, we have $c/b\equiv b/c \bmod pq$, and therefore $c/b\equiv(\frac{q}{p}) \bmod pq$. Thus $a/b=(a/c)(c/b)\equiv(\frac{p}{q})(\frac{q}{p}) \bmod pq$, as desired. □

## 3.2 Finite fields

We recall some standard facts about finite fields. For each prime power $q$ there is, up to isomorphism, a unique field $\mathbb{F}_q$ with $q$ elements (and it is easy to show that the order of every finite field is a prime power). We have the prime fields $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$, and for any positive integer $n$ the field $\mathbb{F}_{p^n}$ can be constructed as the splitting field of the (separable) polynomial $x^{p^n} - x$ over $\mathbb{F}_p$ (thus every finite field is a Galois extension of its prime field). More generally, every degree $n$ extension of $\mathbb{F}_q$ is isomorphic to $\mathbb{F}_{q^n}$, the splitting field of $x^{q^n} - x$ over $\mathbb{F}_q$, and the Galois group $\operatorname{Gal}(\mathbb{F}_{q^n}/\mathbb{F}_q)$ is cyclic over order $n$, generated by the $q$-power Frobenius automorphism $x \mapsto x^q$. We have the inclusion $\mathbb{F}_{q^m} \subseteq \mathbb{F}_{q^n}$ if and only if $m$ divides $n$: if $m|n$ then $x^{q^m} = x$ implies $x^{q^n} = x$, and if $\mathbb{F}_{q^m} \subseteq \mathbb{F}_{q^n}$ then $\mathbb{F}_{q^n}$ has dimension $n/m$ as a vector space over $\mathbb{F}_{q^m}$.

While defining $\mathbb{F}_q = \mathbb{F}_{p^n}$ as a splitting field is conceptually simple, in practice we typically represent $\mathbb{F}_q$ more explicitly by adjoining the root of an irreducible polynomial $f \in \mathbb{F}_p[x]$ of degree $n$ and define $\mathbb{F}_q$ as the ring quotient $\mathbb{F}_p[x]/(f)$. The ring $\mathbb{F}_p[x]$ is a principle ideal domain, so the prime ideal $(f)$ is maximal and the quotient is therefore a field. Such an irreducible polynomial always exists: by the primitive element theorem we know that the separable extension $\mathbb{F}_q/\mathbb{F}_p$ can be constructed as $\mathbb{F}_p(\alpha)$ for some $\alpha \in \mathbb{F}_q$ whose minimal polynomial $f \in \mathbb{F}_p[x]$ is irreducible and of degree $n$. While no deterministic polynomial time algorithm is known for constructing $f$ (even for $n = 2$ (!)), in practice the problem is readily solved using a randomized algorithm, as discussed below.

Elements $s$ and $t$ of $\mathbb{F}_q \simeq \mathbb{F}_p[x]/(f)$ correspond to polynomials in $\mathbb{F}_p[x]$ of degree at most $n$. The sum $s+t$ is computed as in $\mathbb{F}_p[x]$, and the product $st$ is computed as a product in $\mathbb{F}_p[x]$ and then reduced modulo $f$, using Euclidean division and taking the remainder. To compute the inverse of $s$, one uses the (extended) Euclidean gcd algorithm to compute polynomials $u, v \in \mathbb{F}_p[x]$ that satisfy

$$us + vf = \gcd(s, f) = 1,$$

and $u$ is then the inverse of $s$ modulo $f$; note that $\gcd(s, f) = 1$ since $f$ is irreducible. Using fast algorithms for polynomial arithmetic, all of the field operations in $\mathbb{F}_q$ can be computed in time that is quasi-linear in $\log q = n \log p$, which is also the amount of space needed to represent an element of $\mathbb{F}_q$ (up to a constant factor).

**Example 3.2.** $\mathbb{F}_8 \simeq \mathbb{F}_2[t]/(t^3 + t + 1) = \{0, 1, t, t+1, t^2, t^2+1, t^2+t, t^2+t+1\}$ is a finite field of order 8 in which, for example, $(t^2+1)(t^2+t) = t+1$. Note that $\mathbb{F}_2 = \{0, 1\}$ is its only proper subfield (in particular, $\mathbb{F}_4 \not\subseteq \mathbb{F}_8$).

The most thing we need to know about finite fields is that their multiplicative groups are cyclic. This is an immediate consequence of a more general fact.

**Theorem 3.3.** *Any finite subgroup $G$ of the multiplicative group of a field $k$ is cyclic.*

*Proof.* The group $G$ must be abelian, so by the structure theorem for finite abelian groups it is isomorphic to a product of cyclic groups

$$G \simeq \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z},$$

where each $n_i > 1$ and we may assume that $n_i | n_{i+1}$. If $G$ is not cyclic, then $k \geq 2$ and $G$ contains a subgroup isomorphic to $\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_1\mathbb{Z}$ and therefore contains at least $n_1^2 > n_1$ elements whose orders divide $n_1$. But the polynomial $x^{n_1} - 1$ has at most $n_1$ roots in $k$, so this is not possible and $G$ must be cyclic. □

## 3.3 Rational points on conics over finite fields

We now turn to the problem of finding rational points on conics over finite fields. We begin by proving that, unlike the situation over $\mathbb{Q}$, there is always a rational point to find.

**Theorem 3.4.** *Let $C/\mathbb{F}_q$ be a conic over a finite field of odd characteristic. Then $C$ has a rational point.*

*Proof.* As shown in Lecture 2, by completing the square we can put $C$ in the form $ax^2 + by^2 + cz^2 = 0$. If any of $a, b, c$ is zero, say $c$, then $(0 : 0 : 1)$ is a rational point on $C$, so we now assume otherwise. The group $\mathbb{F}_q^\times$ is cyclic and has even order $q - 1$, so it contains exactly $\frac{q-1}{2}$ squares. Therefore the set $S = \{y^2 : y \in \mathbb{F}_q\}$ has cardinality $\frac{q+1}{2}$ (since it also includes 0), as does the set $T = \{-by^2 - c : y \in \mathbb{F}_q\}$, since it is a linear transformation of $S$. Similarly, the set $U = \{ax^2 : x \in \mathbb{F}_q\}$ has cardinality $\frac{q+1}{2}$. The sets $T$ and $U$ cannot be disjoint, since the sum of their cardinalities is larger than $\mathbb{F}_q$, so we must have some $-by_0^2 - c \in T$ equal to some $ax_0^2 \in U$, and $(x_0 : y_0 : 1)$ is then a rational point on $C$. $\square$

**Corollary 3.5.** *Let $C/\mathbb{F}_q$ be a conic over a finite field. Then one of the following holds*

1. *$C$ is geometrically irreducible, isomorphic to $\mathbb{P}^1$, and has $q + 1$ rational points.*
2. *$C$ is reducible over $\mathbb{F}_q$, isomorphic to the union of two rational projective lines, and has $2q + 1$ rational points.*
3. *$C$ is reducible over $\mathbb{F}_q^2$, but not over $\mathbb{F}_q$, isomorphic over $\mathbb{F}_q^2$ to the union of two projective lines with a single rational point at their intersection.*

*In every case we have $\#C(\mathbb{F}_q) \equiv 1 \bmod q$.*

*Proof.* If $C$ is geometrically irreducible then we are in case 1 and the conclusion follows from Theorem 2.3, since we know by Theorem 3.4 that $C$ has a rational point. Otherwise, $C$ must be the product of two degree 1 curves (projective lines), which must intersect at at a single point. If the lines can be defined over $\mathbb{F}_q$ then we are in case 2 and have $2(q+1)-1 = 2q+1$ projective points and otherwise the lines must be defined over the quadratic extension $\mathbb{F}_{q^2}$. which is case 3. The non-trivial element of the Galois group $\mathrm{Gal}(\mathbb{F}_{q^2}/\mathbb{F}_q)$ swaps the two lines and must fix their intersection, which consequently lies in $\mathbb{F}_q$. $\square$

**Remark 3.6.** Theorem 3.4 and Corollary 3.5 also hold in characteristic 2.

## 3.4 Root finding

Let $f$ be a univariate polynomial over a finite field $\mathbb{F}_q$. We now consider the problem of how to find the roots of $f$ that lie in $\mathbb{F}_q$. This will allow us, in particular, to compute the square root of an element $a \in \mathbb{F}_q$ by taking $f(x) = x^2 - a$, which is a necessary ingredient for finding rational points on conics over $\mathbb{F}_q$, and also over $\mathbb{Q}$. Recall that the critical step of the descent algorithm we saw in Lecture 2 for finding a rational point on a conic over $\mathbb{Q}$ required us to compute square roots modulo a square-free integer $n$; this is achieved by computing square roots modulo each of the prime factors of $n$ and applying the Chinese remainder theorem (of course this requires us to compute the prime factorization of $n$, which is actually the hard part).

No deterministic polynomial-time algorithm is know for root-finding over finite fields. Indeed, even the special case of computing square roots modulo a prime is not known to have a deterministic polynomial-time solution.[1] But if we are prepared to use randomized algorithms (which we are), we can quite solve this problem quite efficiently. The algorithm we give here was originally proposed by Berlekamp for prime fields [1], and then refined and extended by Rabin [3], whose presentation we follow here. This algorithm is a great example of how randomness can be exploited in a number-theoretic setting. As we will see, it is quite efficient, with an expected running time that is quasi-quadratic in the size of the input.

### 3.4.1 Randomized algorithms

Randomized algorithms are typically classified as one of two types: *Monte Carlo* or *Las Vegas*. Monte Carlo algorithms are randomized algorithms whose output may be incorrect, depending on random choices made by the algorithm, but whose running time is bounded by a function of its input size, independent of any random choices. The probability of error is required to be less than $1/2 - \epsilon$, for some $\epsilon > 0$, and can be made arbitrarily small be running the algorithm repeatedly and using the output that occurs most often. In contrast, a Las Vegas algorithm always produces a correct output, but its running time may depend on random choices made by the algorithm and need not be bounded as a function of the input size (but we do require its expected running time to be finite). As a trivial example, consider an algorithm to compute $a + b$ that first flips a coin repeatedly until it gets a head and then computes $a + b$ and outputs the result. The running time of this algorithm may be arbitrarily long, even when computing $1 + 1 = 2$, but its *expected* running time is $O(n)$, where $n$ is the size of the inputs (typically measured in bits).

Las Vegas algorithms are generally preferred, particularly in mathematical applications, where we generally require provably correct results. Note that any Monte Carlo algorithm whose output can be verified can always be converted to a Las Vegas algorithm (just run the algorithm repeatedly until you get an answer that is verifiably correct). The root-finding algorithm we present here is of the Las Vegas type.

### 3.4.2 Factoring with gcds

The roots of our polynomial $f \in \mathbb{F}_q[x]$ all lie in the algebraic closure $\overline{\mathbb{F}}_q$. The roots that actually lie in $\mathbb{F}_q$ are distinguished by the fact that they are fixed by the Frobenius automorphism $x \mapsto x^q$. It follows that the roots of $f$ that lie in $\mathbb{F}_q$ are precisely those that are also roots of the polynomial $x^q - x$. Thus the polynomial

$$g = \gcd(f, x^q - x)$$

has the form $\prod_i (x - \alpha_i)$, where the $\alpha_i$ range over the distinct roots of $f$ that lie in $\mathbb{F}_q$. If $f$ has no roots in $\mathbb{F}_q$ then $g$ will have degree 0, and otherwise we can reduce the problem of finding a root of $f$ to the problem of finding a root of $g$, a polynomial whose roots are distinct and known to lie in $\mathbb{F}_q$. Note that this already gives us a deterministic algorithm to determine whether or not $f$ actually has any roots in $\mathbb{F}_q$, but in order to actually find one we may need to factor $g$, and this is where we will use a randomized approach.

In order to compute $\gcd(f, x^q - x)$ efficiently, one does *not* compute $x^q - x$ and then take the gcd with $f$; this would take time exponential in $\log q$, whereas we want an algorithm whose running time is polynomial in the *size* of $f$, which is proportional to $\deg f \log q$.

[1] If one assumes the extended Riemann Hypothesis, this and many other special cases of the root-finding problem can be solved in polynomial time.

Instead, one computes $x^q \bmod f$ by exponentiating the polynomial $x$ in the ring $\mathbb{F}_q[x]/(f)$, whose elements are uniquely represented by polynomials of degree less than $d = \deg f$. Each multiplication in this ring involves the computation of a product in $\mathbb{F}_q[x]$ followed by a reduction modulo $f$. This reduction is achieved using Euclidean division, and can be accomplished within a constant factor of the time required by the multiplication. The computation of $x^q$ is achieved using binary exponentiation (or some other efficient method of exponentiation), where one performs a sequence of squarings and multiplications by $x$ based on the binary representation of $q$, and requires just $O(\log q)$ multiplications in $\mathbb{F}_q[x](f)$. Once we have computed $x^q \bmod f$, we subtract $x$ and compute $g = \gcd(f, x^q - x)$.

Assuming that $q$ is odd (which we do), we may factor the polynomial $x^q - x$ as

$$x^q - x = x(x^s - 1)(x^s + 1),$$

where $s = (q-1)/2$. Ignoring the root 0 (which we can easily check separately), this factorization splits $\mathbb{F}_q^\times$ precisely in half: the roots of $x^s - 1$ are the elements of $\mathbb{F}_q^\times$ that are quadratic residues, and the roots of $x^s + 1$ are the elements of $\mathbb{F}_q^\times$ that are not. If we compute

$$h = \gcd(g, x^s - 1),$$

we obtain a divisor of $g$ whose roots are precisely the roots of $g$ that are quadratic residues. If we suppose that the roots of $g$ are as likely as not to be quadratic residues, we should expect the degree of $h$ to be approximately half the degree of $g$, and so long as the degree of $h$ is strictly between 0 and $\deg g$, one of $h$ or $g/h$ is a polynomial of degree at most half the degree of $g$ and whose roots are all roots of our original polynomial $f$.

To make further progress, and to obtain an algorithm that is guaranteed to work no matter how the roots of $g$ are distributed in $\mathbb{F}_q$, we take a randomized approach. Rather than using the fixed polynomial $x^s - 1$, we consider random polynomials of the form

$$(x + \delta)^s - 1,$$

where $\delta$ is uniformly distributed over $\mathbb{F}_q$. We claim that if $\alpha$ and $\beta$ are any two nonzero roots of $g$, then with probability 1/2, exactly one of these is a root $(x+\delta)^s - 1$. It follows from this claim that so long as $g$ has at least 2 distinct nonzero roots, the probability that the polynomial $h = \gcd(g, (x+\delta)^s + 1)$ is a proper divisor of $g$ is at least 1/2.

Let us say that two elements $\alpha, \beta \in \mathbb{F}_q$ are of *different type* if they are both nonzero and $\alpha^s \neq \beta^s$. Our claim is an immediate consequence of the following theorem from [3].

**Theorem 3.7** (Rabin). *For every pair of distinct $\alpha, \beta \in \mathbb{F}_q$ we have*

$$\#\{\delta \in \mathbb{F}_q : \alpha + \delta \textit{ and } \beta + \delta \textit{ are of different type}\} = \frac{q-1}{2}.$$

*Proof.* Consider the map $\phi(\delta) = \frac{\alpha+\delta}{\beta+\delta}$, defined for $\delta \neq -\beta$. We claim that $\phi$ is a bijection form the set $\mathbb{F}_q \backslash \{-\beta\}$ to the set $\mathbb{F}_q \backslash \{1\}$. The sets are the same size, so we just need to show surjectivity. Let $\gamma \in \mathbb{F}_q - \{1\}$, then we wish to find a solution $x \neq -\beta$ to $\gamma = \frac{\alpha+x}{\beta+x}$. We have $\gamma(\beta + x) = \alpha + x$ which means $x - \gamma x = \gamma\beta - \alpha$. This yields $x = \frac{\gamma\beta-\alpha}{1-\gamma}$, which is not equal to $-\beta$, since $\alpha \neq \beta$. Thus $\phi$ is surjective.

We now note that

$$\phi(\delta)^s = \frac{(\alpha+\delta)^s}{(\beta+\delta)^s}$$

is $-1$ if and only if $\alpha + \delta$ and $\beta + \delta$ are of different type. The elements $\gamma = \phi(\delta)$ for which $\gamma^s = -1$ are precisely the non-residues in $\mathbb{F}_q \backslash \{1\}$, of which there are exactly $(q-1)/2$. □

We now give the algorithm.

```
Algorithm FindRoot($f$)
Input: A polynomial $f \in \mathbb{F}_q[x]$.
Output: An element $r \in \mathbb{F}_q$ such that $f(r) = 0$, or null if no such $r$ exists.

1. If $f(0) = 0$ then return 0.
2. Compute $g = \gcd(f, x^q - x)$.
3. If $\deg g = 0$ then return null.
4. While $\deg g > 1$:
    a. Pick a random $\delta \in \mathbb{F}_q$.
    b. Compute $h = \gcd(g, (x + \delta)^s - 1)$.
    c. If $0 < \deg h < \deg g$ then replace $g$ by $h$ or $g/h$, whichever has lower degree.
5. Return $r = -b/a$, where $g(x) = ax + b$.
```

It is clear that the output of the algorithm is always correct, since every root of the polynomial $g$ computed in step 2 is a root of $f$, and when $g$ is updated in step 4c it is always replaced by a proper divisor. We now consider its complexity.

It follows from Theorem 3.7 that the polynomial $h$ computed in step 4b is a proper divisor of $g$ with probability at least $1/2$, since $g$ has at least two distinct nonzero roots $\alpha, \beta \in \mathbb{F}_q$. Thus the expected number of iterations needed to obtain a proper factor $h$ of $g$ is bounded by 2. The degree of $h$ is at most half the degree of $g$, and the total cost of computing all the polynomials $h$ during step 4 is actually within a constant factor the cost of computing $g$ in step 2.

Using fast algorithms for multiplications and the gcd computation, the time to compute $g$ can be bounded by

$$O(\mathsf{M}(d \log q)(\log q + \log d)$$

bit operations, where $\mathsf{M}(b)$ denotes the time to multiply to $b$-bit integers and is asymptotically bounded by $\mathsf{M}(b) = O(b \log b \log \log b)$ (in fact one can do slightly better). The details of this complexity analysis and the efficient implementation of finite field arithmetic will not concern us in this course, we refer the reader to [2] for a comprehensive treatment, or see these notes for a brief overview. The key point is that this time complexity is polynomial in $d \log q$, in fact it is essentially quadratic, and in practice we can quite quickly find roots of polynomials even over very large finite fields. same complexity bound, and the total expected running time is $O(\mathsf{M}(nd)(n + \log d))$.

The algorithm can easily be modified to find all the distinct roots of $f$, by modifying step 4c to recursively find the roots of both $h$ and $g/h$, this only increases the running time by a factor of $O(\log d)$. Assuming that $d$ is less than the charcteristic of $\mathbb{F}_q$, one can easily determine the multiplicity of each root of $f$: a root $\alpha$ of $f$ occurs with multiplicity $k$ if and only if $\alpha$ is a root of $f^{(k)}$ but not a root of $f^{(k+1)}$, where $f^{(k)}$ denotes the $k$th derivative of $f$. The time to perform this computation is negligible compared to the time to find the distinct roots.

## 3.5 Finding rational points on curves over finite fields

Now that we know how to find roots of univariate polynomials in finite fields (and in particular, square roots), we can easily find a rational point on any conic over a finite field (and enumerate all the rational points if we wish). As above, let us assume $\mathbb{F}_q$ has odd characteristic, so we can put our conic $C$ is diagonal form $x^2 + by^2 + cz^2 = 0$. If $C$ is geometrically reducible then, as proved on Problem Set 1, it is singular and one of $a, b, c$ must be 0. So one of $(1 : 0 : 0)$, $(0 : 1 : 0)$, $(0 : 0 : 1)$ is a rational point on the curve, and in the case that $C$ is reducible over $\mathbb{F}_q$ we can determine the equations of the two lines whose union forms $C$ by computing square roots in $\mathbb{F}_q$; for example, if $c = 0$ we can compute $ax^2 + by^2 = (\sqrt{a}x + \sqrt{-b}y)(\sqrt{a}x + \sqrt{-b}y)$. It is then straight-forward to enumerate all the rational points on $C$.

Now let us suppose that $C$ is geometrically irreducible, in which case we must have $abc \neq 0$. If any of $-a/b, -b/c, -c/a$ is a square in $\mathbb{F}_q$, then we can find a rational point with one coordinate equal to 0 by computing a square-root. Otherwise we know that every rational point $(x_0, y_0, z_0) \in C(\mathbb{F}_q)$ satisfies $x_0y_0z_0 \neq 0$, so we can assume $z_0 = 1$. For each of the $q - 1$ possible nonzero choices for $y_0$, we get either 0 or 2 rational points on $C$, depending on whether $-(by_0^2 + c)/a$ is a square or not. By Corollary .refcor:ffconicpts, We know there are a total of $q + 1$ rational points, so for exactly $(q + 1)/2$ values of $y_0$ we must have $-(by_0^2 + c)/a$ square. Thus if we pick $y_0 \in \mathbb{F}_q$ at random, we have a better than 50/50 chance of finding a rational point on $C$ by computing $\sqrt{-(by_0^2 + c)/a}$. This gives us a Las Vegas algorithm for finding a rational point on $C$ whose expected running time is within a constant factor of the time to compute a square-root in $\mathbb{F}_q$, which is quasi-quadratic in $\log q$. Once we have a rational point on our irreducible conic $C$, we can enumerate them all using the parameterization we computed in Lecture 2.

**Remark 3.8.** The argument above applies more generally. Suppose we have a geometrically irreducible plane curve $C$ defined by a homogeneous polynomial $f(x, y, z)$ of some fixed degree $d$ It follows from the Hasse-Weil bounds, which we will see later in course, that $\#C(\mathbb{F}_q) = q + O(\sqrt{q})$. Assuming $q \gg d$, if we pick a random projective pair $(y_0 : z_0)$ and then attempt to find a root $x_0$ of the univariate polynomial $g(x) = f(x, y_0, z_0)$, we will succeed with a probability that asymptotically approaches $1/d$ as $q \to \infty$. This yields a Las Vegas algorithm for finding a rational point on $C$ in time quasi-quadratic in $\log q$.

## References

[1] Elwyn R. Berlekamp, *Factoring polynomials over large finite fields*, Mathematics of Computation **24** (1970), 713–735.

[2] Joachim von zur Gathen and Jürgen Garhard, *Modern Computer Algebra*, third edition, Cambridge University Press, 2013.

[3] Michael O. Rabin, *Probabilistic algorithms in finite fields*, SIAM Journal of Computing **9** (1980), 273–280.

[4] G. Rousseau, *On the quadratic reciprocity law*, Journal of the Australian Mathematical Society (Series A) **51** (1991), 423–425.