12

Quadratic reciprocity and computing modular square roots

In §2.8, we initiated an investigation of quadratic residues. This chapter continues this investigation. Recall that an integer a is called a quadratic residue modulo a positive integer n if $\gcd(a, n) = 1$ and $a \equiv b^2 \pmod{n}$ for some integer b.

First, we derive the famous law of quadratic reciprocity. This law, while historically important for reasons of pure mathematical interest, also has important computational applications, including a fast algorithm for testing if an integer is a quadratic residue modulo a prime.

Second, we investigate the problem of computing modular square roots: given a quadratic residue a modulo n, compute an integer b such that $a \equiv b^2 \pmod{n}$. As we will see, there are efficient probabilistic algorithms for this problem when n is prime, and more generally, when the factorization of n into primes is known.

12.1 The Legendre symbol

For an odd prime p and an integer a with $\gcd(a, p) = 1$, the **Legendre symbol** $(a \mid p)$ is defined to be 1 if a is a quadratic residue modulo p, and -1 otherwise. For completeness, one defines $(a \mid p) = 0$ if $p \mid a$. The following theorem summarizes the essential properties of the Legendre symbol.

Theorem 12.1. *Let p be an odd prime, and let $a, b \in \mathbb{Z}$. Then we have:*

(i) $(a \mid p) \equiv a^{(p-1)/2} \pmod{p}$*; in particular,* $(-1 \mid p) = (-1)^{(p-1)/2}$*;*

(ii) $(a \mid p)(b \mid p) = (ab \mid p)$*;*

(iii) $a \equiv b \pmod{p}$ *implies* $(a \mid p) = (b \mid p)$*;*

(iv) $(2 \mid p) = (-1)^{(p^2-1)/8}$*;*

(v) if q is an odd prime, then $(p \mid q) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}(q \mid p)$.

Part (i) of the theorem is just a restatement of Euler's criterion (Theorem 2.21).

As was observed in Theorem 2.31, this implies that -1 is a quadratic residue modulo p if and only if $p \equiv 1 \pmod 4$. Thus, the quadratic residuosity of -1 modulo p is determined by the residue class of p modulo 4.

Part (ii) of the theorem follows immediately from part (i), and part (iii) is an immediate consequence of the definition of the Legendre symbol.

Part (iv), which we will prove below, can also be recast as saying that 2 is a quadratic residue modulo p if and only if $p \equiv \pm 1 \pmod 8$. Thus, the quadratic residuosity of 2 modulo p is determined by the residue class of p modulo 8.

Part (v), which we will also prove below, is the **law of quadratic reciprocity**. Note that when $p = q$, both $(p \mid q)$ and $(q \mid p)$ are zero, and so the statement of part (v) is trivially true—the interesting case is when $p \neq q$, and in this case, part (v) is equivalent to saying that

$$(p \mid q)(q \mid p) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}.$$

Thus, the Legendre symbols $(p \mid q)$ and $(q \mid p)$ have the same values if and only if either $p \equiv 1 \pmod 4$ or $q \equiv 1 \pmod 4$. As the following examples illustrate, this result also shows that for a given odd prime q, the quadratic residuosity of q modulo another odd prime p is determined by the residue class of p modulo either q or $4q$.

Example 12.1. Let us characterize those primes p modulo which 5 is a quadratic residue. Since $5 \equiv 1 \pmod 4$, the law of quadratic reciprocity tells us that $(5 \mid p) = (p \mid 5)$. Now, among the numbers ± 1, ± 2, the quadratic residues modulo 5 are ± 1. It follows that 5 is a quadratic residue modulo p if and only if $p \equiv \pm 1 \pmod 5$. This example obviously generalizes, replacing 5 by any prime $q \equiv 1 \pmod 4$, and replacing the above congruences modulo 5 by appropriate congruences modulo q. □

Example 12.2. Let us characterize those primes p modulo which 3 is a quadratic residue. Since $3 \not\equiv 1 \pmod 4$, we must be careful in our application of the law of quadratic reciprocity. First, suppose that $p \equiv 1 \pmod 4$. Then $(3 \mid p) = (p \mid 3)$, and so 3 is a quadratic residue modulo p if and only if $p \equiv 1 \pmod 3$. Second, suppose that $p \not\equiv 1 \pmod 4$. Then $(3 \mid p) = -(p \mid 3)$, and so 3 is a quadratic residue modulo p if and only if $p \equiv -1 \pmod 3$. Putting this all together, we see that 3 is quadratic residue modulo p if and only if

$$p \equiv 1 \pmod 4 \text{ and } p \equiv 1 \pmod 3$$

or

$$p \equiv -1 \pmod 4 \text{ and } p \equiv -1 \pmod 3.$$

Using the Chinese remainder theorem, we can restate this criterion in terms of

residue classes modulo 12: 3 is quadratic residue modulo p if and only if $p \equiv \pm 1 \pmod{12}$. This example obviously generalizes, replacing 3 by any prime $q \equiv -1 \pmod 4$, and replacing the above congruences modulo 12 by appropriate congruences modulo $4q$. □

The rest of this section is devoted to a proof of parts (iv) and (v) of Theorem 12.1. The proof is completely elementary, although a bit technical.

Theorem 12.2 (Gauss' lemma). *Let p be an odd prime and let a be an integer not divisible by p. Define $\alpha_j := ja \bmod p$ for $j = 1, \ldots, (p-1)/2$, and let n be the number of indices j for which $\alpha_j > p/2$. Then $(a \mid p) = (-1)^n$.*

Proof. Let $r_1, \ldots, r_n$ denote the values α_j that exceed $p/2$, and let $s_1, \ldots, s_k$ denote the remaining values α_j. The r_i's and s_i's are all distinct and non-zero. We have $0 < p - r_i < p/2$ for $i = 1, \ldots, n$, and no $p - r_i$ is an s_j; indeed, if $p - r_i = s_j$, then $s_j \equiv -r_i \pmod p$, and writing $s_j = ua \bmod p$ and $r_i = va \bmod p$, for some $u, v = 1, \ldots, (p-1)/2$, we have $ua \equiv -va \pmod p$, which implies $u \equiv -v \pmod p$, which is impossible.

It follows that the sequence of numbers $s_1, \ldots, s_k, p - r_1, \ldots, p - r_n$ is just a reordering of $1, \ldots, (p-1)/2$. Then we have

$$\begin{aligned}((p-1)/2)! &\equiv s_1 \cdots s_k(-r_1) \cdots (-r_n) \\ &\equiv (-1)^n s_1 \cdots s_k r_1 \cdots r_n \\ &\equiv (-1)^n((p-1)/2)!\, a^{(p-1)/2} \pmod p,\end{aligned}$$

and canceling the factor $((p-1)/2)!$, we obtain $a^{(p-1)/2} \equiv (-1)^n \pmod p$, and the result follows from the fact that $(a \mid p) \equiv a^{(p-1)/2} \pmod p$. □

Theorem 12.3. *If p is an odd prime and $\gcd(a, 2p) = 1$, then $(a \mid p) = (-1)^t$ where $t = \sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor$. Also, $(2 \mid p) = (-1)^{(p^2-1)/8}$.*

Proof. Let a be an integer not divisible by p, but which may be even, and let us adopt the same notation as in the statement and proof of Theorem 12.2; in particular, $\alpha_1, \ldots, \alpha_{(p-1)/2}$, $r_1, \ldots, r_n$, and $s_1, \ldots, s_k$ are as defined there. Note that $ja = p\lfloor ja/p \rfloor + \alpha_j$, for $j = 1, \ldots, (p-1)/2$, so we have

$$\sum_{j=1}^{(p-1)/2} ja = \sum_{j=1}^{(p-1)/2} p\lfloor ja/p \rfloor + \sum_{j=1}^{n} r_j + \sum_{j=1}^{k} s_j. \tag{12.1}$$

Moreover, as we saw in the proof of Theorem 12.2, the sequence of numbers

$s_1, \ldots, s_k, p - r_1, \ldots, p - r_n$ is a reordering of $1, \ldots, (p-1)/2$, and hence

$$\sum_{j=1}^{(p-1)/2} j = \sum_{j=1}^{n} (p - r_j) + \sum_{j=1}^{k} s_j = np - \sum_{j=1}^{n} r_j + \sum_{j=1}^{k} s_j. \tag{12.2}$$

Subtracting (12.2) from (12.1), we get

$$(a-1) \sum_{j=1}^{(p-1)/2} j = p\Big(\sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor - n \Big) + 2 \sum_{j=1}^{n} r_j. \tag{12.3}$$

Note that

$$\sum_{j=1}^{(p-1)/2} j = \frac{p^2 - 1}{8}, \tag{12.4}$$

which together with (12.3) implies

$$(a-1)\frac{p^2-1}{8} \equiv \sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor - n \pmod 2. \tag{12.5}$$

If a is odd, (12.5) implies

$$n \equiv \sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor \pmod 2. \tag{12.6}$$

If $a = 2$, then $\lfloor 2j/p \rfloor = 0$ for $j = 1, \ldots, (p-1)/2$, and (12.5) implies

$$n \equiv \frac{p^2 - 1}{8} \pmod 2. \tag{12.7}$$

The theorem now follows from (12.6) and (12.7), together with Theorem 12.2. □

Note that this last theorem proves part (iv) of Theorem 12.1. The next theorem proves part (v).

Theorem 12.4. *If p and q are distinct odd primes, then*

$$(p \mid q)(q \mid p) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}.$$

Proof. Let S be the set of pairs of integers (x, y) with $1 \le x \le (p-1)/2$ and $1 \le y \le (q-1)/2$. Note that S contains no pair (x, y) with $qx = py$, so let us partition S into two subsets: S_1 contains all pairs (x, y) with $qx > py$, and S_2 contains all pairs (x, y) with $qx < py$. Note that $(x, y) \in S_1$ if and only if

$1 \le x \le (p-1)/2$ and $1 \le y \le \lfloor qx/p \rfloor$. So $|S_1| = \sum_{x=1}^{(p-1)/2} \lfloor qx/p \rfloor$. Similarly, $|S_2| = \sum_{y=1}^{(q-1)/2} \lfloor py/q \rfloor$. So we have

$$\frac{p-1}{2}\frac{q-1}{2} = |S| = |S_1| + |S_2| = \sum_{x=1}^{(p-1)/2} \lfloor qx/p \rfloor + \sum_{y=1}^{(q-1)/2} \lfloor py/q \rfloor,$$

and Theorem 12.3 implies

$$(p \mid q)(q \mid p) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}. \quad \square$$

EXERCISE 12.1. Characterize those odd primes p for which $(15 \mid p) = 1$, in terms of the residue class of p modulo 60.

EXERCISE 12.2. Let p be an odd prime. Show that the following are equivalent:

(a) $(-2 \mid p) = 1$;

(b) $p \equiv 1$ or $3 \pmod 8$;

(c) $p = r^2 + 2t^2$ for some $r, t \in \mathbb{Z}$.

12.2 The Jacobi symbol

Let a, n be integers, where n is positive and odd, so that $n = q_1 \cdots q_k$, where the q_i's are odd primes, not necessarily distinct. Then the **Jacobi symbol** $(a \mid n)$ is defined as

$$(a \mid n) := (a \mid q_1) \cdots (a \mid q_k),$$

where $(a \mid q_i)$ is the Legendre symbol. By definition, $(a \mid 1) = 1$ for all $a \in \mathbb{Z}$. Thus, the Jacobi symbol essentially extends the domain of definition of the Legendre symbol. Note that $(a \mid n) \in \{0, \pm 1\}$, and that $(a \mid n) = 0$ if and only if $\gcd(a, n) > 1$. The following theorem summarizes the essential properties of the Jacobi symbol.

Theorem 12.5. *Let m, n be odd, positive integers, and let $a, b \in \mathbb{Z}$. Then we have:*

(i) $(ab \mid n) = (a \mid n)(b \mid n)$;

(ii) $(a \mid mn) = (a \mid m)(a \mid n)$;

(iii) $a \equiv b \pmod n$ *implies* $(a \mid n) = (b \mid n)$;

(iv) $(-1 \mid n) = (-1)^{(n-1)/2}$;

(v) $(2 \mid n) = (-1)^{(n^2-1)/8}$;

(vi) $(m \mid n) = (-1)^{\frac{m-1}{2}\frac{n-1}{2}}(n \mid m)$.

Proof. Parts (i)–(iii) follow directly from the definition (exercise).

For parts (iv) and (vi), one can easily verify (exercise) that for all odd integers $n_1, \ldots, n_k$,

$$\sum_{i=1}^{k}(n_i - 1)/2 \equiv (n_1 \cdots n_k - 1)/2 \pmod 2.$$

Part (iv) easily follows from this fact, along with part (ii) of this theorem and part (i) of Theorem 12.1 (exercise). Part (vi) easily follows from this fact, along with parts (i) and (ii) of this theorem, and part (v) of Theorem 12.1 (exercise).

For part (v), one can easily verify (exercise) that for odd integers $n_1, \ldots, n_k$,

$$\sum_{i=1}^{k}(n_i^2 - 1)/8 \equiv (n_1^2 \cdots n_k^2 - 1)/8 \pmod 2.$$

Part (v) easily follows from this fact, along with part (ii) of this theorem, and part (iv) of Theorem 12.1 (exercise). □

As we shall see later, this theorem is extremely useful from a computational point of view — with it, one can efficiently compute $(a \mid n)$, without having to know the prime factorization of either a or n. Also, in applying this theorem it is useful to observe that for all odd integers m, n,

- $(-1)^{(n-1)/2} = 1 \iff n \equiv 1 \pmod 4$;
- $(-1)^{(n^2-1)/8} = 1 \iff n \equiv \pm 1 \pmod 8$;
- $(-1)^{((m-1)/2)((n-1)/2)} = 1 \iff m \equiv 1 \pmod 4$ or $n \equiv 1 \pmod 4$.

Suppose a is a quadratic residue modulo n, so that $a \equiv b^2 \pmod n$, where $\gcd(a, n) = 1 = \gcd(b, n)$. Then by parts (iii) and (i) of Theorem 12.5, we have $(a \mid n) = (b^2 \mid n) = (b \mid n)^2 = 1$. Thus, if a is a quadratic residue modulo n, then $(a \mid n) = 1$. The converse, however, does not hold: $(a \mid n) = 1$ does *not* imply that a is a quadratic residue modulo n (see Exercise 12.3 below).

It is sometimes useful to view the Jacobi symbol as a group homomorphism. Let n be an odd, positive integer. Define the **Jacobi map**

$$\begin{aligned} J_n : \quad & \mathbb{Z}_n^* \to \{\pm 1\} \\ & [a]_n \mapsto (a \mid n). \end{aligned}$$

First, we note that by part (iii) of Theorem 12.5, this definition is unambiguous. Second, we note that since $\gcd(a, n) = 1$ implies $(a \mid n) = \pm 1$, the image of J_n is indeed contained in $\{\pm 1\}$. Third, we note that by part (i) of Theorem 12.5, J_n is a group homomorphism. Since J_n is a group homomorphism, it follows that its kernel, $\operatorname{Ker} J_n$, is a subgroup of $\mathbb{Z}_n^*$.

EXERCISE 12.3. Let n be an odd, positive integer, and consider the Jacobi map J_n.

(a) Show that $(\mathbb{Z}_n^*)^2 \subseteq \operatorname{Ker} J_n$.

(b) Show that if n is the square of an integer, then $\operatorname{Ker} J_n = \mathbb{Z}_n^*$.

(c) Show that if n is not the square of an integer, then $[\mathbb{Z}_n^* : \operatorname{Ker} J_n] = 2$ and $[\operatorname{Ker} J_n : (\mathbb{Z}_n^*)^2] = 2^{r-1}$, where r is the number of distinct prime divisors of n.

EXERCISE 12.4. Let p and q be distinct primes, with $p \equiv q \equiv 3 \pmod 4$, and let $n := pq$.

(a) Show that $[-1]_n \in \operatorname{Ker} J_n \setminus (\mathbb{Z}_n^*)^2$, and from this, conclude that the cosets of $(\mathbb{Z}_n^*)^2$ in $\operatorname{Ker} J_n$ are the two distinct cosets $(\mathbb{Z}_n^*)^2$ and $[-1]_n(\mathbb{Z}_n^*)^2$.

(b) Let $\delta \in \mathbb{Z}_n^* \setminus \operatorname{Ker} J_n$. Show that the map from $\{0,1\} \times \{0,1\} \times (\mathbb{Z}_n^*)^2$ to $\mathbb{Z}_n^*$ that sends (a, b, γ) to $\delta^a(-1)^b\gamma$ is a bijection.

12.3 Computing the Jacobi symbol

Suppose we are given an odd, positive integer n, along with an integer a, and we want to compute the Jacobi symbol $(a \mid n)$. Theorem 12.5 suggests the following algorithm:

```
σ ← 1
repeat
    // loop invariant: n is odd and positive
    a ← a mod n
    if a = 0 then
        if n = 1 then return σ else return 0
    compute a′, h such that a = 2^h a′ and a′ is odd
    if h ≢ 0 (mod 2) and n ≢ ±1 (mod 8) then σ ← −σ
    if a′ ≢ 1 (mod 4) and n ≢ 1 (mod 4) then σ ← −σ
    (a, n) ← (n, a′)
forever
```

That this algorithm correctly computes the Jacobi symbol $(a \mid n)$ follows directly from Theorem 12.5. Using an analysis similar to that of Euclid's algorithm, one easily sees that the running time of this algorithm is $O(\text{len}(a)\,\text{len}(n))$.

EXERCISE 12.5. Develop a "binary" Jacobi symbol algorithm, that is, one that uses only addition, subtractions, and "shift" operations, analogous to the binary gcd algorithm in Exercise 4.6.

EXERCISE 12.6. This exercise develops a probabilistic primality test based on the Jacobi symbol. For odd integer $n > 1$, define

$$G_n := \{\alpha \in \mathbb{Z}_n^* : \alpha^{(n-1)/2} = J_n(\alpha)\},$$

where $J_n : \mathbb{Z}_n^* \to \{\pm 1\}$ is the Jacobi map.

(a) Show that G_n is a subgroup of $\mathbb{Z}_n^*$.

(b) Show that if n is prime, then $G_n = \mathbb{Z}_n^*$.

(c) Show that if n is composite, then $G_n \subsetneq \mathbb{Z}_n^*$.

(d) Based on parts (a)–(c), design and analyze an efficient probabilistic primality test that works by choosing a random, non-zero element $\alpha \in \mathbb{Z}_n$, and testing if $\alpha \in G_n$.

12.4 Testing quadratic residuosity

In this section, we consider the problem of testing whether a is a quadratic residue modulo n, for given integers a and n, from a computational perspective.

12.4.1 Prime modulus

For an odd prime p, we can test if an integer a is a quadratic residue modulo p by either performing the exponentiation $a^{(p-1)/2} \bmod p$ or by computing the Legendre symbol $(a \mid p)$. Assume that $0 \leq a < p$. Using a standard repeated squaring algorithm, the former method takes time $O(\text{len}(p)^3)$, while using the Euclidean-like algorithm of the previous section, the latter method takes time $O(\text{len}(p)^2)$. So clearly, the latter method is to be preferred.

12.4.2 Prime-power modulus

For an odd prime p, we know that a is a quadratic residue modulo p^e if and only if a is a quadratic residue modulo p (see Theorem 2.30). So this case immediately reduces to the previous one.

12.4.3 Composite modulus

For odd, composite n, if we know the factorization of n, then we can also determine if a is a quadratic residue modulo n by determining if it is a quadratic residue modulo each prime divisor p of n (see Exercise 2.39). However, without knowledge of this factorization (which is in general believed to be hard to compute), there is no efficient algorithm known. We can compute the Jacobi symbol $(a \mid n)$; if this

is -1 or 0, we can conclude that a is not a quadratic residue; otherwise, we cannot conclude much of anything.

12.5 Computing modular square roots

In this section, we consider the problem of computing a square root of a modulo n, given integers a and n, where a is a quadratic residue modulo n.

12.5.1 Prime modulus

Let p be an odd prime, and let a be an integer such that $0 < a < p$ and $(a \mid p) = 1$. We would like to compute a square root of a modulo p. Let $\alpha := [a]_p \in \mathbb{Z}_p^*$, so that we can restate our problem as that of finding $\beta \in \mathbb{Z}_p^*$ such that $\beta^2 = \alpha$, given $\alpha \in (\mathbb{Z}_p^*)^2$.

We first consider the special case where $p \equiv 3 \pmod 4$, in which it turns out that this problem can be solved very easily. Indeed, we claim that in this case

$$\beta := \alpha^{(p+1)/4}$$

is a square root of α—note that since $p \equiv 3 \pmod 4$, the number $(p+1)/4$ is an integer. To show that $\beta^2 = \alpha$, suppose $\alpha = \tilde{\beta}^2$ for some $\tilde{\beta} \in \mathbb{Z}_p^*$. We know that there is such a $\tilde{\beta}$, since we are assuming that $\alpha \in (\mathbb{Z}_p^*)^2$. Then we have

$$\beta^2 = \alpha^{(p+1)/2} = \tilde{\beta}^{p+1} = \tilde{\beta}^2 = \alpha,$$

where we used Fermat's little theorem for the third equality. Using a repeated-squaring algorithm, we can compute β in time $O(\mathrm{len}(p)^3)$.

Now we consider the general case, where we may have $p \not\equiv 3 \pmod 4$. Here is one way to efficiently compute a square root of α, assuming we are given, in addition to α, an auxiliary input $\gamma \in \mathbb{Z}_p^* \setminus (\mathbb{Z}_p^*)^2$ (how one obtains such a γ is discussed below).

Let us write $p-1 = 2^h m$, where m is odd. For every $\delta \in \mathbb{Z}_p^*$, δ^m has multiplicative order dividing 2^h. Since $\alpha^{2^{h-1}m} = 1$, α^m has multiplicative order dividing 2^{h-1}. Since $\gamma^{2^{h-1}m} = -1$, γ^m has multiplicative order precisely 2^h. Since there is only one subgroup of $\mathbb{Z}_p^*$ of order 2^h, it follows that γ^m generates this subgroup, and that $\alpha^m = \gamma^{mx}$ for some integer x, where $0 \leq x < 2^h$ and x is even. We can find x by computing the discrete logarithm of α^m to the base γ^m, using the algorithm in §11.2.3. Setting $\kappa = \gamma^{mx/2}$, we have

$$\kappa^2 = \alpha^m.$$

We are not quite done, since we now have a square root of α^m, and not of α.

Since m is odd, we may write $m = 2t + 1$ for some non-negative integer t. It then follows that

$$(\kappa\alpha^{-t})^2 = \kappa^2\alpha^{-2t} = \alpha^m\alpha^{-2t} = \alpha^{m-2t} = \alpha.$$

Thus, $\kappa\alpha^{-t}$ is a square root of α.

Let us summarize the above algorithm for computing a square root of $\alpha \in (\mathbb{Z}_p^*)^2$, assuming we are given $\gamma \in \mathbb{Z}_p^* \setminus (\mathbb{Z}_p^*)^2$, in addition to α:

compute positive integers m, h such that $p - 1 = 2^h m$ with m odd
$\gamma' \leftarrow \gamma^m$, $\alpha' \leftarrow \alpha^m$
compute $x \leftarrow \log_{\gamma'} \alpha'$ *// note that* $0 \le x < 2^h$ *and* x *is even*
$\beta \leftarrow (\gamma')^{x/2}\alpha^{-\lfloor m/2 \rfloor}$
output β

The work done outside the discrete logarithm calculation amounts to just a handful of exponentiations modulo p, and so takes time $O(\mathrm{len}(p)^3)$. The time to compute the discrete logarithm is $O(h\,\mathrm{len}(h)\,\mathrm{len}(p)^2)$. So the total running time of this procedure is

$$O(\mathrm{len}(p)^3 + h\,\mathrm{len}(h)\,\mathrm{len}(p)^2).$$

The above procedure assumed we had at hand a non-square γ. If $h = 1$, which means that $p \equiv 3 \pmod 4$, then $(-1 \mid p) = -1$, and so we are done. However, we have already seen how to efficiently compute a square root in this case.

If $h > 1$, we can find a non-square γ using a probabilistic search algorithm. Simply choose γ at random, test if it is a square, and if so, repeat. The probability that a random element of $\mathbb{Z}_p^*$ is a square is $1/2$; thus, the expected number of trials until we find a non-square is 2; moreover, the running time per trial is $O(\mathrm{len}(p)^2)$, and hence the expected running time of this probabilistic search algorithm is $O(\mathrm{len}(p)^2)$.

12.5.2 Prime-power modulus

Let p be an odd prime, let a be an integer relatively prime to p, and let $e > 1$ be an integer. We know that a is a quadratic residue modulo p^e if and only if a is a quadratic residue modulo p. Suppose that a is a quadratic residue modulo p, and that we have found an integer b such that $b^2 \equiv a \pmod p$, using, say, one of the procedures described in §12.5.1. From this, we can easily compute a square root of a modulo p^e using the following technique, which is known as **Hensel lifting**.

More generally, suppose that for some $f \ge 1$, we have computed an integer b satisfying the congruence $b^2 \equiv a \pmod{p^f}$, and we want to find an integer c satisfying the congruence $c^2 \equiv a \pmod{p^{f+1}}$. Clearly, if $c^2 \equiv a \pmod{p^{f+1}}$, then

$c^2 \equiv a \pmod{p^f}$, and so $c \equiv \pm b \pmod{p^f}$. So let us set $c = b + p^f h$, and solve for h. We have

$$c^2 \equiv (b + p^f h)^2 \equiv b^2 + 2bp^f h + p^{2f} h^2 \equiv b^2 + 2bp^f h \pmod{p^{f+1}}.$$

So we want to find an integer h satisfying the linear congruence

$$2bp^f h \equiv a - b^2 \pmod{p^{f+1}}. \tag{12.8}$$

Since $p \nmid 2b$, we have $\gcd(2bp^f, p^{f+1}) = p^f$. Furthermore, since $b^2 \equiv a \pmod{p^f}$, we have $p^f \mid (a - b^2)$. Therefore, Theorem 2.5 implies that (12.8) has a unique solution h modulo p, which we can efficiently compute as in Example 4.3.

By iterating the above procedure, starting with a square root of a modulo p, we can quickly find a square root of a modulo p^e. We leave a detailed analysis of the running time of this procedure to the reader.

12.5.3 Composite modulus

To find square roots modulo n, where n is an odd composite modulus, if we know the prime factorization of n, then we can use the above procedures for finding square roots modulo primes and prime powers, and then use the algorithm of the Chinese remainder theorem to get a square root modulo n.

However, if the factorization of n is not known, then there is no efficient algorithm known for computing square roots modulo n. In fact, one can show that the problem of finding square roots modulo n is at least as hard as the problem of factoring n, in the sense that if there is an efficient algorithm for computing square roots modulo n, then there is an efficient (probabilistic) algorithm for factoring n.

We now present an algorithm to factor n, using a modular square-root algorithm A as a subroutine. For simplicity, we assume that A is deterministic, and that for all n and for all $\alpha \in (\mathbb{Z}_n^*)^2$, $A(n, \alpha)$ outputs a square root of α. Also for simplicity, we shall assume that n is of the form $n = pq$, where p and q are distinct, odd primes. In Exercise 12.15 below, you are asked to relax these restrictions. Our algorithm runs as follows:

$\beta \xleftarrow{\$} \mathbb{Z}_n^+$, $d \leftarrow \gcd(\mathrm{rep}(\beta), n)$
if $d > 1$ then
    output d
else
    $\alpha \leftarrow \beta^2$, $\beta' \leftarrow A(n, \alpha)$
    if $\beta = \pm\beta'$
        then output "failure"
        else output $\gcd(\mathrm{rep}(\beta - \beta'), n)$

Here, $\mathbb{Z}_n^+$ denotes the set of non-zero elements of $\mathbb{Z}_n$. Also, recall that $\mathrm{rep}(\beta)$ denotes the canonical representative of β.

First, we argue that the algorithm outputs either "failure" or a non-trivial factor of n. Clearly, if $\beta \notin \mathbb{Z}_n^*$, then the value d computed by the algorithm is a non-trivial factor. So suppose $\beta \in \mathbb{Z}_n^*$. In this case, the algorithm invokes A on inputs n and $\alpha := \beta^2$, obtaining a square root β' of α. Suppose that $\beta \neq \pm\beta'$, and set $\gamma := \beta - \beta'$. What we need to show is that $\gcd(\mathrm{rep}(\gamma), n)$ is a non-trivial factor of n. To see this, consider the ring isomorphism of the Chinese remainder theorem

$$\begin{aligned} \theta : \quad \mathbb{Z}_n &\to \mathbb{Z}_p \times \mathbb{Z}_q \\ [a]_n &\mapsto ([a]_p, [a]_q). \end{aligned}$$

Suppose $\theta(\beta') = (\beta'_1, \beta'_2)$. Then the four square roots of α are

$$\beta' = \theta^{-1}(\beta'_1, \beta'_2),\ -\beta' = \theta^{-1}(-\beta'_1, -\beta'_2),\ \theta^{-1}(-\beta'_1, \beta'_2),\ \theta^{-1}(\beta'_1, -\beta'_2).$$

The assumption that $\beta \neq \pm\beta'$ implies that $\theta(\beta) = (-\beta'_1, \beta'_2)$ or $\theta(\beta) = (\beta'_1, -\beta'_2)$. In the first case, $\theta(\gamma) = (-2\beta'_1, 0)$, which implies $\gcd(\mathrm{rep}(\gamma), n) = q$. In the second case, $\theta(\gamma) = (0, -2\beta'_2)$, which implies $\gcd(\mathrm{rep}(\gamma), n) = p$.

Second, we argue that $\mathsf{P}[\mathcal{F}] \leq 1/2$, where $\mathcal{F}$ is the event that the algorithm outputs "failure." Viewed as a random variable, β is uniformly distributed over $\mathbb{Z}_n^+$. Clearly, $\mathsf{P}[\mathcal{F} \mid \beta \notin \mathbb{Z}_n^*] = 0$. Now consider any fixed $\alpha' \in (\mathbb{Z}_n^*)^2$. Observe that the conditional distribution of β given that $\beta^2 = \alpha'$ is (essentially) the uniform distribution on the set of four square roots of α'. Also observe that the output of A depends only on n and β^2, and so with respect to the conditional distribution given that $\beta^2 = \alpha'$, the output β' of A is fixed. Thus,

$$\mathsf{P}[\mathcal{F} \mid \beta^2 = \alpha'] = \mathsf{P}[\beta = \pm\beta' \mid \beta^2 = \alpha'] = 1/2.$$

Putting everything together, using total probability, we have

$$\begin{aligned} \mathsf{P}[\mathcal{F}] &= \mathsf{P}[\mathcal{F} \mid \beta \notin \mathbb{Z}_n^*]\,\mathsf{P}[\beta \notin \mathbb{Z}_n^*] + \sum_{\alpha' \in (\mathbb{Z}_n^*)^2} \mathsf{P}[\mathcal{F} \mid \beta^2 = \alpha']\,\mathsf{P}[\beta^2 = \alpha'] \\ &= 0 \cdot \mathsf{P}[\beta \notin \mathbb{Z}_n^*] + \sum_{\alpha' \in (\mathbb{Z}_n^*)^2} \frac{1}{2} \cdot \mathsf{P}[\beta^2 = \alpha'] \leq \frac{1}{2}. \end{aligned}$$

Thus, the above algorithm fails to split n with probability at most $1/2$. If we like, we can repeat the algorithm until it succeeds. The expected number of iterations performed will be at most 2.

EXERCISE 12.7. Let p be an odd prime, and let $f \in \mathbb{Z}_p[X]$ be a polynomial with $0 \leq \deg(f) \leq 2$. Design and analyze an efficient, deterministic algorithm that

takes as input p, f, and an element of $\mathbb{Z}_p^* \setminus (\mathbb{Z}_p^*)^2$, and which determines if f has any roots in $\mathbb{Z}_p$, and if so, finds all of the roots. Hint: see Exercise 7.17.

EXERCISE 12.8. Show how to deterministically compute square roots modulo primes $p \equiv 5 \pmod 8$ in time $O(\text{len}(p)^3)$.

EXERCISE 12.9. This exercise develops an alternative algorithm for computing square roots modulo a prime. Let p be an odd prime, let $\beta \in \mathbb{Z}_p^*$, and set $\alpha := \beta^2$. Define $B_\alpha := \{\gamma \in \mathbb{Z}_p : \gamma^2 - \alpha \in (\mathbb{Z}_p^*)^2\}$.

(a) Show that $B_\alpha = \{\gamma \in \mathbb{Z}_p : g(\gamma) = 0\}$, where

$$g := (X - \beta)^{(p-1)/2} - (X + \beta)^{(p-1)/2} \in \mathbb{Z}_p[X].$$

(b) Let $\gamma \in \mathbb{Z}_p \setminus B_\alpha$, and suppose $\gamma^2 \neq \alpha$. Let μ, ν be the uniquely determined elements of $\mathbb{Z}_p$ satisfying the polynomial congruence

$$\mu + \nu X \equiv (\gamma - X)^{(p-1)/2} \pmod{X^2 - \alpha}.$$

Show that $\mu = 0$ and $\nu^{-2} = \alpha$.

(c) Using parts (a) and (b), design and analyze a probabilistic algorithm that computes a square root of a given $\alpha \in (\mathbb{Z}_p^*)^2$ in expected time $O(\text{len}(p)^3)$.

Note that when $p - 1 = 2^h m$ (m odd), and h is large (e.g., $h \approx \text{len}(p)/2$), the algorithm in the previous exercise is asymptotically faster than the one in §12.5.1; however, the latter algorithm is likely to be faster in practice for the typical case where h is small.

EXERCISE 12.10. Show that the following two problems are deterministic, poly-time equivalent (see discussion just above Exercise 11.10 in §11.3):

(a) Given an odd prime p and $\alpha \in (\mathbb{Z}_p^*)^2$, find $\beta \in \mathbb{Z}_p^*$ such that $\beta^2 = \alpha$.

(b) Given an odd prime p, find an element of $\mathbb{Z}_p^* \setminus (\mathbb{Z}_p^*)^2$.

EXERCISE 12.11. Design and analyze an efficient, deterministic algorithm that takes as input primes p and q, such that $q \mid (p - 1)$, along with an element $\alpha \in \mathbb{Z}_p^*$, and determines whether or not $\alpha \in (\mathbb{Z}_p^*)^q$.

EXERCISE 12.12. Design and analyze an efficient, deterministic algorithm that takes as input primes p and q, such that $q \mid (p - 1)$ but $q^2 \nmid (p - 1)$, along with an element $\alpha \in (\mathbb{Z}_p^*)^q$, and computes a qth root of α, that is, an element $\beta \in \mathbb{Z}_p^*$ such that $\beta^q = \alpha$.

EXERCISE 12.13. Design and analyze an algorithm that takes as input primes p and q, such that $q \mid (p - 1)$, along with an element $\alpha \in (\mathbb{Z}_p^*)^q$, and computes a qth root of α. (Unlike Exercise 12.12, we now allow $q^2 \mid (p - 1)$.) Your algorithm may

be probabilistic, and should have an expected running time that is bounded by $q^{1/2}$ times a polynomial in $\mathrm{len}(p)$. Hint: Exercise 4.13 may be useful.

EXERCISE 12.14. Let p be an odd prime, γ be a generator for $\mathbb{Z}_p^*$, and α be any element of $\mathbb{Z}_p^*$. Define

$$B(p,\gamma,\alpha) := \begin{cases} 1 & \text{if } \log_\gamma \alpha \geq (p-1)/2; \\ 0 & \text{if } \log_\gamma \alpha < (p-1)/2. \end{cases}$$

Suppose that there is an algorithm that efficiently computes $B(p,\gamma,\alpha)$ for all p,γ,α as above. Show how to use this algorithm as a subroutine in an efficient, probabilistic algorithm that computes $\log_\gamma \alpha$ for all p,γ,α as above. Hint: in addition to the algorithm that computes B, use algorithms for testing quadratic residuosity and computing square roots modulo p, and "read off" the bits of $\log_\gamma \alpha$ one at a time.

EXERCISE 12.15. Suppose there is a probabilistic algorithm A that takes as input a positive integer n, and an element $\alpha \in (\mathbb{Z}_n^*)^2$. Assume that for all n, and for a randomly chosen $\alpha \in (\mathbb{Z}_n^*)^2$, A computes a square root of α with probability at least 0.001. Here, the probability is taken over the random choice of α and the random choices of A. Show how to use A to construct another probabilistic algorithm A' that takes n as input, runs in expected polynomial time, and that satisfies the following property:

> for all n, A' outputs the complete factorization of n into primes with probability at least 0.999.

EXERCISE 12.16. Suppose there is a probabilistic algorithm A that takes as input positive integers n and m, and an element $\alpha \in (\mathbb{Z}_n^*)^m$. It outputs either "failure," or an mth root of α. Furthermore, assume that A runs in expected polynomial time, and that for all n and m, and for randomly chosen $\alpha \in (\mathbb{Z}_n^*)^m$, A succeeds in computing an mth root of α with probability $\varepsilon(n,m)$. Here, the probability is taken over the random choice of α, as well as the random choices made during the execution of A. Show how to use A to construct another probabilistic algorithm A' that takes as input n, m, and $\alpha \in (\mathbb{Z}_n^*)^m$, runs in expected polynomial time, and that satisfies the following property:

> if $\varepsilon(n,m) \geq 0.001$, then *for all* $\alpha \in (\mathbb{Z}_n^*)^m$, A' computes an mth root of α with probability at least 0.999.

12.6 The quadratic residuosity assumption

Loosely speaking, the **quadratic residuosity (QR)** assumption is the assumption that it is hard to distinguish squares from non-squares in $\mathbb{Z}_n^*$, where n is of the form

$n = pq$, and p and q are distinct primes. This assumption plays an important role in cryptography. Of course, since the Jacobi symbol is easy to compute, for this assumption to make sense, we have to restrict our attention to elements of $\operatorname{Ker} J_n$, where $J_n : \mathbb{Z}_n^* \to \{\pm 1\}$ is the Jacobi map. We know that $(\mathbb{Z}_n^*)^2 \subseteq \operatorname{Ker} J_n$ (see Exercise 12.3). Somewhat more precisely, the QR assumption is the assumption that it is hard to distinguish a random element in $\operatorname{Ker} J_n \setminus (\mathbb{Z}_n^*)^2$ from a random element in $(\mathbb{Z}_n^*)^2$, given n (but not its factorization!).

To give a rough idea as to how this assumption may be used in cryptography, assume that $p \equiv q \equiv 3 \pmod 4$, so that $[-1]_n \in \operatorname{Ker} J_n \setminus (\mathbb{Z}_n^*)^2$, and moreover, $\operatorname{Ker} J_n \setminus (\mathbb{Z}_n^*)^2 = [-1]_n(\mathbb{Z}_n^*)^2$ (see Exercise 12.4). The value n can be used as a public key in a public-key cryptosystem (see §4.7). Alice, knowing the public key, can encrypt a single bit $b \in \{0, 1\}$ as $\beta := (-1)^b \alpha^2$, where Alice chooses $\alpha \in \mathbb{Z}_n^*$ at random. The point is, if $b = 0$, then β is uniformly distributed over $(\mathbb{Z}_n^*)^2$, and if $b = 1$, then β is uniformly distributed over $\operatorname{Ker} J_n \setminus (\mathbb{Z}_n^*)^2$. Now Bob, knowing the secret key, which is the factorization of n, can easily determine if $\beta \in (\mathbb{Z}_n^*)^2$ or not, and hence deduce the value of the encrypted bit b. However, under the QR assumption, an eavesdropper, seeing just n and β, cannot effectively figure out what b is.

Of course, the above scheme is much less efficient than the RSA cryptosystem presented in §4.7, but nevertheless, has attractive properties; in particular, its security is very closely tied to the QR assumption, whereas the security of RSA is a bit less well understood.

EXERCISE 12.17. Suppose that A is a probabilistic algorithm that takes as input n of the form $n = pq$, where p and q are distinct primes such that $p \equiv q \equiv 3 \pmod 4$. The algorithm also takes as input $\alpha \in \operatorname{Ker} J_n$, and outputs either 0 or 1. Furthermore, assume that A runs in expected polynomial time. Define two random variables, X_n and Y_n, as follows: X_n is defined to be the output of A on input n and a value α chosen at random from $\operatorname{Ker} J_n \setminus (\mathbb{Z}_n^*)^2$, and Y_n is defined to be the output of A on input n and a value α chosen at random from $(\mathbb{Z}_n^*)^2$. In both cases, the value of the random variable is determined by the random choice of α, as well as the random choices made by the algorithm. Define $\varepsilon(n) := |\mathsf{P}[X_n = 1] - \mathsf{P}[Y_n = 1]|$. Show how to use A to design a probabilistic, expected polynomial time algorithm A' that takes as input n as above and $\alpha \in \operatorname{Ker} J_n$, and outputs either "square" or "non-square," with the following property:

> if $\varepsilon(n) \geq 0.001$, then *for all* $\alpha \in \operatorname{Ker} J_n$, the probability that A' correctly identifies whether $\alpha \in (\mathbb{Z}_n^*)^2$ is at least 0.999.

Hint: use the Chernoff bound.

EXERCISE 12.18. Assume the same notation as in the previous exercise. Define the random variable X'_n to be the output of A on input n and a value α chosen at random from $\operatorname{Ker} J_n$. Show that $|\mathsf{P}[X'_n = 1] - \mathsf{P}[Y_n = 1]| = \varepsilon(n)/2$. Thus, the problem of distinguishing $\operatorname{Ker} J_n$ from $(\mathbb{Z}_n^*)^2$ is essentially equivalent to the problem of distinguishing $\operatorname{Ker} J_n \setminus (\mathbb{Z}_n^*)^2$ from $(\mathbb{Z}_n^*)^2$.

12.7 Notes

The proof we present here of Theorem 12.1 is essentially the one from Niven and Zuckerman [72]. Our proof of Theorem 12.5 follows closely the one found in Bach and Shallit [11].

Exercise 12.6 is based on Solovay and Strassen [99].

The probabilistic algorithm in §12.5.1 can be made deterministic under a generalization of the Riemann hypothesis. Indeed, as discussed in §10.5, under such a hypothesis, Bach's result [10] implies that the least positive integer that is not a quadratic residue modulo p is at most $2 \log p$ (this follows by applying Bach's result with the subgroup $(\mathbb{Z}_p^*)^2$ of $\mathbb{Z}_p^*$). Thus, we may find the required element $\gamma \in \mathbb{Z}_p^* \setminus (\mathbb{Z}_n^*)^2$ in deterministic polynomial time, just by brute-force search. The best *unconditional* bound on the smallest positive integer that is not a quadratic residue modulo p is due to Burgess [22], who gives a bound of $p^{\alpha+o(1)}$, where $\alpha := 1/(4\sqrt{e}) \approx 0.15163$.

Goldwasser and Micali [41] introduced the quadratic residuosity assumption to cryptography (as discussed in §12.6). This assumption has subsequently been used as the basis for numerous cryptographic schemes.