3.8 The Index Calculus Method for Computing Discrete Logarithms in $\mathbb{F}_p$

The *index calculus* is a method for solving the discrete logarithm problem in a finite field $\mathbb{F}_p$. The algorithm uses smooth numbers and bears some similarity to the sieve methods that we have studied in this chapter, which is why we cover it here, rather than in Chap. 2, where we originally discussed discrete logarithms.

The idea behind the index calculus is fairly simple. We want to solve the discrete logarithm problem

$$g^x \equiv h \pmod{p}, \tag{3.27}$$

where the prime p and the integers g and h are given. For simplicity, we will assume that g is a primitive root modulo p, so its powers give all of $\mathbb{F}_p^*$.

Rather than solving (3.27) directly, we instead choose a value B and solve the discrete logarithm problem

$$g^x \equiv \ell \pmod{p} \qquad \text{for all primes } \ell \le B.$$

In other words, we compute the discrete logarithm $\log_g(\ell)$ for every prime satisfying $\ell \le B$.

Having done this, we next look at the quantities

$$h \cdot g^{-k} \pmod{p} \qquad \text{for } k = 1, 2, \ldots$$

until we find a value of k such that $h \cdot g^{-k} \pmod{p}$ is B-smooth. For this value of k we have

$$h \cdot g^{-k} \equiv \prod_{\ell \le B} \ell^{e_\ell} \pmod{p} \tag{3.28}$$

for certain exponents e_ℓ. We rewrite (3.28) in terms of discrete logarithms as

$$\log_g(h) \equiv k + \sum_{\ell \le B} e_\ell \cdot \log_g(\ell) \pmod{p-1}, \tag{3.29}$$

where recall that discrete logarithms are defined only modulo $p-1$. But we are assuming that we already computed $\log_g(\ell)$ for all primes $\ell \le B$. Hence (3.29) gives the value of $\log_g(h)$.

It remains to explain how to find $\log_g(\ell)$ for small primes ℓ. Again the idea is simple. For a random selection of exponents i we compute

$$g_i \equiv g^i \pmod{p} \qquad \text{with } 0 < g_i < p.$$

If g_i is not B-smooth, then we discard it, while if g_i is B-smooth, then we can factor it as

$$g_i = \prod_{\ell \le B} \ell^{u_\ell(i)}.$$

In terms of discrete logarithms, this gives the relation

$$i \equiv \log_g(g_i) \equiv \sum_{\ell \le B} u_\ell(i) \cdot \log_g(\ell) \pmod{p-1}. \tag{3.30}$$

Notice that the only unknown quantities in the formula (3.30) are the discrete logarithm values $\log_g(\ell)$. So if we can find more than $\pi(B)$ equations like (3.30), then we can use linear algebra to solve for the $\log_g(\ell)$ "variables."

This method of solving the discrete logarithm problem in $\mathbb{F}_p$ is called the *index calculus*, where recall from Sect. 2.2 that *index* is an older name for *discrete logarithm*. The index calculus first appears in work of Western and Miller [148] in 1968, so it predates by a few years the invention of public key cryptography. The method was independently rediscovered by several cryptographers in the 1970s after the publication of the Diffie–Hellman paper [38].

Remark 3.57. A minor issue that we have ignored is the fact that the linear equations (3.30) are congruences modulo $p-1$. Standard linear algebra methods such as Gaussian elimination do not work well modulo composite numbers, because there are numbers that do not have multiplicative inverses. The Chinese remainder theorem (Theorem 2.24) solves this problem. First we solve the congruences (3.30) modulo q for each prime q dividing $p-1$. Then, if q appears in the factorization of $p-1$ to a power q^e, we lift the solution from $\mathbb{Z}/q\mathbb{Z}$ to $\mathbb{Z}/q^e\mathbb{Z}$. Finally, we use the Chinese remainder theorem to combine solutions modulo prime powers to obtain a solution modulo $p-1$. In cryptographic applications one should choose p such that $p-1$ is divisible by a large prime; otherwise, the Pohlig–Hellman algorithm (Sect. 2.9) solves the discrete logarithm problem. For example, if we select $p = 2q+1$ with q prime, then the index calculus requires us to solve simultaneous congruences (3.30) modulo q and modulo 2.

There are many implementation issues that arise and tricks that have been developed in practical applications of the index calculus. We do not pursue these matters here, but are content to present a small numerical example illustrating how the index calculus works.

Example 3.58. We let p be the prime $p = 18443$ and use the index calculus to solve the discrete logarithm problem

$$37^x \equiv 211 \pmod{18443}.$$

We note that $g = 37$ is a primitive root modulo $p = 18443$ We take $B = 5$, so our factor base is the set of primes $\{2, 3, 5\}$. We start by taking random powers of $g = 37$ modulo 18443 and pick out the ones that are B-smooth. A couple of hundred attempts gives four equations:

$$\begin{aligned} g^{12708} &\equiv 2^3 \cdot 3^4 \cdot 5 \pmod{18443}, & g^{11311} &\equiv 2^3 \cdot 5^2 \pmod{18443}, \\ g^{15400} &\equiv 2^3 \cdot 3^3 \cdot 5 \pmod{18443}, & g^{2731} &\equiv 2^3 \cdot 3 \cdot 5^4 \pmod{18443}. \end{aligned} \tag{3.31}$$

These in turn give linear relations for the discrete logarithms of 2, 3, and 5 to the base g. For example, the first one says that

$$12708 = 3 \cdot \log_g(2) + 4 \cdot \log_g(3) + \log_g(5).$$

To ease notation, we let

$$x_2 = \log_g(2), \quad x_3 = \log_g(3), \quad \text{and} \quad x_5 = \log_g(5).$$

Then the four congruences (3.31) become the following four linear relations:

$$\begin{aligned} 12708 &= 3x_2 + 4x_3 + x_5 \pmod{18442}, \\ 11311 &= 3x_2 + 2x_5 \pmod{18442}, \\ 15400 &= 3x_2 + 3x_3 + x_5 \pmod{18442}, \\ 2731 &= 3x_2 + x_3 + 4x_5 \pmod{18442}. \end{aligned} \tag{3.32}$$

Note that the formulas (3.32) are congruences modulo

$$p - 1 = 18442 = 2 \cdot 9221,$$

since discrete logarithms are defined only modulo $p - 1$. The number 9221 is prime, so we need to solve the system of linear equations (3.32) modulo 2 and modulo 9221. This is easily accomplished by Gaussian elimination, i.e., by adding multiples of one equation to another to eliminate variables. The solutions are

$$\begin{aligned} (x_2, x_3, x_5) &\equiv (1, 0, 1) \pmod{2}, \\ (x_2, x_3, x_5) &\equiv (5733, 6529, 6277) \pmod{9221}. \end{aligned}$$

Combining these solutions yields

$$(x_2, x_3, x_5) \equiv (5733, 15750, 6277) \pmod{18442}.$$

We check the solutions by computing

$$37^{5733} \equiv 2 \pmod{18443}, \quad 37^{15750} \equiv 3 \pmod{18443}, \quad 37^{6277} \equiv 5 \pmod{18443}.$$

Recall that our ultimate goal is to solve the discrete logarithm problem

$$37^x \equiv 211 \pmod{18443}.$$

We compute the value of $211 \cdot 37^{-k} \pmod{18443}$ for random values of k until we find a value that is B-smooth. After a few attempts we find that

$$211 \cdot 37^{-9549} \equiv 2^5 \cdot 3^2 \cdot 5^2 \pmod{18443}.$$

Using the values of the discrete logs of 2, 3, and 5 from above, this yields

$$\begin{aligned} \log_g(211) &= 9549 + 5\log_g(2) + 2\log_g(3) + 2\log_g(5) \\ &= 9549 + 5 \cdot 5733 + 2 \cdot 15750 + 2 \cdot 6277 \equiv 8500 \pmod{18442}. \end{aligned}$$

Finally, we check our answer $\log_g(211) = 8500$ by computing

$$37^{8500} \equiv 211 \pmod{18443}. \quad \checkmark$$

Remark 3.59. We can roughly estimate the running time of the index calculus as follows. Using a factor base consisting of primes less than B, we need to find approximately $\pi(B)$ numbers of the form $g^i \pmod p$ that are B-smooth. Proposition 3.48 suggests that we should take $B = L(p)^{1/\sqrt{2}}$, and then we will have to check approximately $L(p)^{\sqrt{2}}$ values of i. There is also the issue of checking each value to see whether it is B-smooth, but sieve-type methods can be used to speed the process. Further, using ideas based on the number field sieve, the running time can be further reduced to a small power $L_{1/3}(p)$. In any case, the index calculus is a subexponential algorithm for solving the discrete logarithm problem in $\mathbb{F}_p^*$. This stands in marked contrast to the discrete logarithm problem in elliptic curve groups, which we study in Chap. 6. Currently, the best known algorithms to solve the general discrete logarithm problem in elliptic curve groups are fully exponential.

3.9 Quadratic Residues and Quadratic Reciprocity

Let p be a prime number. Here is a simple mathematical question:

> How can Bob tell whether a given number a is equal to a square modulo p?

For example, suppose that Alice asks Bob whether 181 is a square modulo 1223. One way for Bob to answer Alice's question is by constructing a table of squares modulo 1223 as illustrated in Table 3.8, but this is a lot of work, so he gave up after computing $96^2 \bmod 1223$. Alice picked up the computation where Bob stopped and eventually found that $437^2 \equiv 181 \pmod{1223}$. Thus the answer to her question is that 181 is indeed a square modulo 1223. Similarly, if Alice is sufficiently motivated to continue the table all the way up to $1222^2 \bmod 1223$, she can verify that the number 385 is not a square modulo 1223, because it does not appear in her table. (In fact, Alice can save half her time by computing only up to $611^2 \bmod 1223$, since a^2 and $(p-a)^2$ have the same values modulo p.)

Our goal in this section is to describe a more much efficient way to check if a number is a square modulo a prime. We begin with a definition.

Definition. Let p be an odd prime number and let a be a number with $p \nmid a$. We say that a is a *quadratic residue modulo* p if a is a square modulo p, i.e., if there is a number c so that $c^2 \equiv a \pmod p$. If a is not a square modulo p, i.e., if there exists no such c, then a is called a *quadratic nonresidue modulo* p.

Example 3.60. The numbers 968 and 1203 are both quadratic residues modulo 1223, since

$$453^2 \equiv 968 \pmod{1223} \qquad \text{and} \qquad 375^2 \equiv 1203 \pmod{1223}.$$