best choice turns out to be $m = 17$ (in which case $h = 16510$). The value $q = 2^{127}$ is a popular choice, because $\#(\mathbf{F}_{2^{127}}^*) = 2^{127} - 1$ is a Mersenne prime.

We now return to the index–calculus algorithm, and describe the final stage. Here we suppose that $y(X) \in \mathbf{F}_q^*$ is the element whose discrete log we wish to compute, and that stage one has already given us the values of $ind(a(X))$ for all $a(X) \in B$. We again choose a random t between 1 and $q-2$, and compute $y_1 = yb^t$, i.e., the unique polynomial $y_1(X) \in \mathbf{F}_p[X]$ of degree $< n$ satisfying $y_1(X) \equiv y(X)b(X)^t \mod f(X)$. As in the first stage of the algorithm, we test whether $y_1(X)$ factors into a constant y_0 times a product of powers of $a(X)$, $a(X) \in B$. If not, we choose another random t, and so on, until we finally have an integer t such that $y_1(X) \equiv y_0 \prod_{a \in B} a(X)^{\alpha_a}$. As soon as this happens, we are done, because $ind(y) = ind(y_1) - t$, by the definition of y_1; and $ind(y_1) = ind(y_0) + \sum \alpha_a ind(a(X))$, in which we know all of the terms on the right. This completes the description of the index–calculus algorithm.

It should be mentioned that in the popular case $p = 2$, an improved method due to D. Coppersmith has significantly speeded up the process of finding discrete logs. For this reason, a discrete log cryptosystem using $\mathbf{F}_{2^n}^*$ is no longer regarded as secure unless n is of the order of 1000. Despite this, these fields $\mathbf{F}_{2^n}$ remain popular because they lend themselves to efficient programming. For a good survey (covering what was known as of 1985), the reader is referred to A. Odlyzko's article (see References below).

If $q = p^n$ is an odd prime power which is k bits long, it turns out that, roughly speaking, the order of magnitude of time needed to solve the discrete log problem in $\mathbf{F}_q^*$ is comparable to what is needed to factor a k-bit integer. That is, from an empirical point of view, the discrete log problem seems to be about as difficult as factoring (though no one has been able to prove a theorem to this effect). In fact, when we discuss factoring algorithms and time estimates for them in the next chapter, we will see that one of the fundamental methods of factoring large integers bears a striking resemblance to the index–calculus algorithm for finding discrete logs.

Thus, at this point it is too early to say whether the public key cryptosystems of the RSA type (based on the difficulty of factoring integers) or the discrete log cryptosystems will eventually prove to be the more secure.

Exercises

Note: Exercises 4, 6, 7(c) and 8 should be attempted only if you have the use of a computer with multiple precision arithmetic programs. (All that is really needed is a program for computing $a^b \mod m$ for very large integers a, b and m; recall that $a^{-1} \mod p$ can be computed by taking a^{p-2}.)

1. If one has occasion to do a lot of arithmetic in a fixed finite field $\mathbf{F}_q$ which is not too large, it can save time first to compose a complete "table of logarithms." In other words, choose a generator g of $\mathbf{F}_q$ and