5.6.3 Dixon's Random Squares Algorithm

Many factoring algorithms are based on the following very simple idea. Suppose we can find $x \not\equiv \pm y \pmod{n}$ such that $x^2 \equiv y^2 \pmod{n}$. Then $n \mid (x-y)(x+y)$ but neither of $x-y$ or $x+y$ is divisible by n. It therefore follows that $\gcd(x+y, n)$ is a non-trivial factor of n (and similarly, $\gcd(x-y, n)$ is also non-trivial factor of n).

As an example, it is easy to verify that $10^2 \equiv 32^2 \pmod{77}$. By computing $\gcd(10+32, 77) = 7$, we discover the factor 7 of 77.

Dixon's algorithm uses a *factor base*, which is a set $\mathcal{B}$ of the b smallest primes, for an appropriate value b. We first obtain several integers z such that all the prime factors of $z^2 \bmod n$ occur in the factor base $\mathcal{B}$. (How this is done will be discussed a bit later.) The idea is to then take the product of several of these z's in such a way that every prime in the factor base is used an even number of times. This then gives us a congruence of the desired type $x^2 \equiv y^2 \pmod{n}$, which (we hope) will lead to a factorization of n.

We illustrate with a carefully contrived example.

Example 5.11 Suppose $n = 15770708441$ (this was the same n that we used in Example 5.9). Let $b = 6$; then $\mathcal{B} = \{2, 3, 5, 7, 11, 13\}$. Consider the three congruences:

$$\begin{aligned} 8340934156^2 &\equiv 3 \times 7 \pmod{n} \\ 12044942944^2 &\equiv 2 \times 7 \times 13 \pmod{n} \\ 2773700011^2 &\equiv 2 \times 3 \times 13 \pmod{n}. \end{aligned}$$

If we take the product of these three congruences, then we have

$$(8340934156 \times 12044942944 \times 2773700011)^2 \equiv (2 \times 3 \times 7 \times 13)^2 \pmod{n}.$$

Reducing the expressions inside the parentheses modulo n, we have

$$9503435785^2 \equiv 546^2 \pmod{n}.$$

Then, using the EUCLIDEAN ALGORITHM, we compute

$$\gcd(9503435785 - 546, 15770708441) = 115759,$$

finding the factor 115759 of n. □

Suppose $\mathcal{B} = \{p_1, \ldots, p_b\}$ is the factor base. Let c be slightly larger than b (say $c = b + 10$), and suppose we have obtained c congruences:

$$z_j{}^2 \equiv p_1{}^{\alpha_{1j}} \times p_2{}^{\alpha_{2j}} \ldots \times p_b{}^{\alpha_{bj}} \pmod{n},$$

$1 \le j \le c$. For each j, consider the vector

$$a_j = (\alpha_{1j} \bmod 2, \ldots, \alpha_{bj} \bmod 2) \in (\mathbb{Z}_2)^b.$$

If we can find a subset of the a_j's that sum modulo 2 to the vector $(0, \ldots, 0)$, then the product of the corresponding z_j's will use each factor in $\mathcal{B}$ an even number of times.

We illustrate by returning to Example 5.11, where there exists a dependence even though $c < b$ in this case.

Example 5.11 (Cont.) The three vectors a_1, a_2, a_3 are as follows:

$$\begin{aligned} a_1 &= (0,1,0,1,0,0) \\ a_2 &= (1,0,0,1,0,1) \\ a_3 &= (1,1,0,0,0,1). \end{aligned}$$

It is easy to see that

$$a_1 + a_2 + a_3 = (0,0,0,0,0,0) \bmod 2.$$

This gives rise to the congruence we saw earlier that successfully factored n. $\square$

Observe that finding a subset of the c vectors $a_1, \ldots, a_c$ that sums modulo 2 to the all-zero vector is nothing more than finding a linear dependence (over $\mathbb{Z}_2$) of these vectors. Provided $c > b$, such a linear dependence must exist, and it can be found easily using the standard method of Gaussian elimination. The reason why we take $c > b + 1$ is that there is no guarantee that any given congruence $x^2 \equiv y^2 \bmod n$ will yield the factorization of n. However, it can be argued that $x \equiv \pm y \pmod{n}$ at most 50% of the time. If $c > b + 1$, then we can obtain several such congruences (arising from different linear dependencies among the a_j's). Hopefully, at least one of the resulting congruences will yield a congruence of the form $x^2 \equiv y^2 \bmod n$ where $x \not\equiv \pm y \pmod{n}$, and a non-trivial factor of n will be obtained.

We now discuss how to obtain integers z such that the values $z^2 \bmod n$ factor completely over a given factor base $\mathcal{B}$. There are several methods of doing this. One way is simply to choose the z's at random; this approach yields the so-called "random squares algorithm". However, it is particularly useful to try integers of the form $j + \lceil\sqrt{kn}\rceil$, $j = 0, 1, 2, \ldots$, $k = 1, 2, \ldots$. These integers tend to be small when squared and reduced modulo n, and hence they have a higher than average probability of factoring over $\mathcal{B}$. Another useful trick is to try integers of the form $z = \lfloor\sqrt{kn}\rfloor$. When squared and reduced modulo n, these integers are a bit less than n.

This means that $-z^2 \bmod n$ is small and can perhaps be factored over $\mathcal{B}$. Alternatively, if we include -1 in $\mathcal{B}$, we can factor $z^2 \bmod n$ over $\mathcal{B}$.

We illustrate these techniques with a small example.

Example 5.12 Suppose that $n = 1829$ and $\mathcal{B} = \{-1, 2, 3, 5, 7, 11, 13\}$. We compute $\sqrt{n} = 42.77$, $\sqrt{2n} = 60.48$, $\sqrt{3n} = 74.07$ and $\sqrt{4n} = 85.53$. Suppose we take $z = 42, 43, 64, 65, 74, 75, 85, 86$. We obtain several following factorizations of $z^2 \bmod n$ over $\mathcal{B}$. In the following table, all congruences are modulo n:

$$\begin{array}{lclcrcl}
{z_1}^2 & \equiv & 42^2 & \equiv & -65 & \equiv & (-1) \times 5 \times 13 \\
{z_2}^2 & \equiv & 43^2 & \equiv & 20 & \equiv & 2^2 \times 5 \\
{z_3}^2 & \equiv & 60^2 & \equiv & -58 & & \\
{z_4}^2 & \equiv & 61^2 & \equiv & 63 & \equiv & 3^2 \times 7 \\
{z_5}^2 & \equiv & 74^2 & \equiv & -11 & \equiv & (-1) \times 11 \\
{z_6}^2 & \equiv & 75^2 & \equiv & 138 & & \\
{z_7}^2 & \equiv & 85^2 & \equiv & -91 & \equiv & (-1) \times 7 \times 13 \\
{z_8}^2 & \equiv & 86^2 & \equiv & 80 & \equiv & 2^4 \times 5.
\end{array}$$

We have six factorizations, which yield six vectors in $(\mathbb{Z}_2)^7$. This is not enough to guarantee a dependence relation, but it turns out to be sufficient in this particular case. The six vectors are as follows:

$$\begin{aligned}
a_1 &= (1, 0, 0, 1, 0, 0, 1) \\
a_2 &= (0, 0, 0, 1, 0, 0, 0) \\
a_4 &= (0, 0, 0, 0, 1, 0, 0) \\
a_5 &= (1, 0, 0, 0, 0, 1, 0) \\
a_7 &= (1, 0, 0, 0, 1, 0, 1) \\
a_8 &= (0, 0, 0, 1, 0, 0, 0).
\end{aligned}$$

Clearly $a_2 + a_8 = (0, 0, 0, 0, 0, 0, 0)$; however, the reader can check that this dependence relation does not yield a factorization of n. A dependence relation that does work is

$$a_1 + a_2 + a_4 + a_7 = (0, 0, 0, 0, 0, 0, 0).$$

The congruence that we obtain is

$$(42 \times 43 \times 61 \times 85)^2 \equiv (2 \times 3 \times 5 \times 7 \times 13)^2 \bmod 1829.$$

This simplifies to give

$$1459^2 \equiv 901^2 \bmod 1829.$$

It is then straightforward to compute

$$\gcd(1459 + 901, 1829) = 59,$$

and thus we have obtained a nontrivial factor of n. □

An important general question is how large the factor base should be (as a function of the integer g that we are attempting to factor) and what the complexity of the algorithm is. In general, there is, of course, a trade-off: if $b = |\mathcal{B}|$ is large, then it is more likely that an integer ${x_j}^2 \bmod n$ factors over $\mathcal{B}$. But the larger b is, the more congruences we need to accumulate before we are able to find a dependence relation. The optimal choice for b can be chosen with the help of some results from number theory. We discuss some of the main ideas now. This will be a heuristic analysis in which we will be assuming that the integers z are chosen randomly.

Suppose that n and m are positive integers. We say that n is *m-smooth* provided that every prime factor of n is less than or equal to m. $\Psi(n, m)$ is defined to be the number of positive integers less than or equal to n that are m-smooth. An important result in number theory says that, if $n \gg m$, then

$$\frac{\Psi(n, m)}{n} \approx \frac{1}{u^u},$$

where $u = \log n / \log m$. Observe that $\Psi(n, m)/n$ represents the probability that a random integer in the set $\{1, \ldots, n\}$ is m-smooth.

Suppose that $n \approx 2^r$ and $m \approx 2^s$. Then $u = \log n / \log m \approx r/s$. Division of an r-bit integer by an s-bit integer can be done in time $O(rs)$. From this, it is not difficult to show that we can determine if an integer in the set $\{1, \ldots, n\}$ is m-smooth in time $O(rsm)$.

Our factor base $\mathcal{B}$ consists of all the primes less than or equal to m. Therefore $|\mathcal{B}| = b = \pi(m) \approx m/\ln m$. We need to find slightly more than b m-smooth squares modulo n in order for the algorithm to succeed. We expect to test bu^u integers in order to find b of them that are m-smooth. Therefore, the expected time to find the necessary m-smooth squares is $O(bu^u \times rsm) = O(rm^2u^u)$. Once this is done, we need to reduce the associated matrix modulo 2, construct our congruence of the form $x^2 \equiv y^2 \pmod{n}$, and apply the EUCLIDEAN ALGORITHM. It can be checked without too much difficulty that these steps can be done in time that is polynomial in r and m.

The total running time can be written in the form $O(r^i m^j u^u)$, where i and j are positive integers. Recall that $n \approx 2^r$ is given, and we are trying to choose $m \approx 2^s$ to optimize the running time. It turns out that the best choice for m is to take $s \approx \sqrt{r \log_2 r}$. Then

$$u \approx \frac{r}{s} \approx \sqrt{\frac{r}{\log_2 r}}.$$

Now we can compute

$$\begin{aligned}
\log_2 u^u &= u \log_2 u \\
&\approx \sqrt{\frac{r}{\log_2 r}} \log_2 \left(\sqrt{\frac{r}{\log_2 r}} \right) \\
&< \sqrt{\frac{r}{\log_2 r}} \log_2 \sqrt{r} \\
&= \sqrt{\frac{r}{\log_2 r}} \times \frac{\log_2 r}{2} \\
&= \frac{\sqrt{r \log_2 r}}{2}.
\end{aligned}$$

Therefore it follows that $u^u \leq 2^{0.5\sqrt{r \log_2 r}}$. It is clear that $m^j \leq 2^{j\sqrt{r \log_2 r}}$, and hence the total running time can be expressed in the form $O\left(2^{c\sqrt{r \log_2 r}}\right)$ for some positive constant c.

Using the fact that $r \approx \log_2 n$, we obtain a running time of $O\left(2^{c\sqrt{\log_2 n \log_2 \log_2 n}}\right)$. Often the running time is expressed in terms of logarithms and exponentials to the base e. A more careful analysis leads to the following commonly stated expected running time:

$$O\left(e^{(1+o(1))\sqrt{\ln n \ln \ln n}}\right).$$