Chapter 9

Computational Number Theory

9.1 The basic groups

We let $\mathbf{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ denote the set of integers. We let $\mathbf{Z}_+ = \{1, 2, \ldots\}$ denote the set of positive integers and $\mathbf{N} = \{0, 1, 2, \ldots\}$ the set of non-negative integers.

9.1.1 Integers mod N

If a, b are integers, not both zero, then their greatest common divisor, denoted $\gcd(a, b)$, is the largest integer d such that d divides a and d divides b. If $\gcd(a, b) = 1$ then we say that a and b are relatively prime. If a, N are integers with $N > 0$ then there are unique integers r, q such that $a = Nq + r$ and $0 \leq r < N$. We call r the remainder upon division of a by N, and denote it by $a \bmod N$. We note that the operation $a \bmod N$ is defined for both negative and non-negative values of a, but only for positive values of N. (When a is negative, the quotient q will also be negative, but the remainder r must always be in the indicated range $0 \leq r < N$.) If a, b are any integers and N is a positive integer, we write $a \equiv b \pmod{N}$ if $a \bmod N = b \bmod N$. We associate to any positive integer N the following two sets:

$$\begin{aligned} \mathbf{Z}_N &= \{0, 1, \ldots, N-1\} \\ \mathbf{Z}_N^* &= \{\, i \in \mathbf{Z} \,:\, 1 \leq i \leq N-1 \text{ and } \gcd(i, N) = 1 \,\} \end{aligned}$$

The first set is called the set of integers mod N. Its size is N, and it contains exactly the integers that are possible values of $a \bmod N$ as a ranges over $\mathbf{Z}$. We define the Euler Phi (or totient) function φ: $\mathbf{Z}_+ \to \mathbf{N}$ by $\varphi(N) = |\mathbf{Z}_N^*|$ for all $N \in \mathbf{Z}_+$. That is, $\varphi(N)$ is the size of the set $\mathbf{Z}_N^*$.

9.1.2 Groups

Let G be a non-empty set, and let $\cdot$ be a binary operation on G. This means that for every two points $a, b \in G$, a value $a \cdot b$ is defined.

Definition 9.1 Let G be a non-empty set and let $\cdot$ denote a binary operation on G. We say that G is a *group* if it has the following properties:

1. CLOSURE: For every $a, b \in G$ it is the case that $a \cdot b$ is also in G.
2. ASSOCIATIVITY: For every $a, b, c \in G$ it is the case that $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
3. IDENTITY: There exists an element $\mathbf{1} \in G$ such that $a \cdot \mathbf{1} = \mathbf{1} \cdot a = a$ for all $a \in G$.
4. INVERTIBILITY: For every $a \in G$ there exists a unique $b \in G$ such that $a \cdot b = b \cdot a = \mathbf{1}$.

The element b in the invertibility condition is referred to as the inverse of the element a, and is denoted a^{-1}. ▌

We now return to the sets we defined above and remark on their group structure. Let N be a positive integer. The operation of addition modulo N takes input any two integers a, b and returns $(a + b) \bmod N$. The operation of multiplication modulo N takes input any two integers a, b and returns $ab \bmod N$.

Fact 9.2 Let N be a positive integer. Then $\mathbf{Z}_N$ is a group under addition modulo N, and $\mathbf{Z}_N^*$ is a group under multiplication modulo N. ▌

In $\mathbf{Z}_N$, the identity element is 0 and the inverse of a is $-a \bmod N = N - a$. In $\mathbf{Z}_N^*$, the identity element is 1 and the inverse of a is a $b \in \mathbf{Z}_N^*$ such that $ab \equiv 1 \pmod{N}$. In may not be obvious why such a b even exists, but it does. We do not prove the above fact here.

In any group, we can define an exponentiation operation which associates to any $a \in G$ and any integer i a group element we denote a^i, defined as follows. If $i = 0$ then a^i is defined to be $\mathbf{1}$, the identity element of the group. If $i > 0$ then

$$a^i = \underbrace{a \cdot a \cdots a}_{i} .$$

If i is negative, then we define $a^i = (a^{-1})^{-i}$. Put another way, let $j = -i$, which is positive, and set

$$a^i = \underbrace{a^{-1} \cdot a^{-1} \cdots a^{-1}}_{j} .$$

With these definitions in place, we can manipulate exponents in the way in which we are accustomed with ordinary numbers. Namely, identities such as the following hold for all $a \in G$ and all $i, j \in \mathbf{Z}$:

$$\begin{aligned} a^{i+j} &= a^i \cdot a^j \\ (a^i)^j &= a^{ij} \\ a^{-i} &= (a^i)^{-1} \\ a^{-i} &= (a^{-1})^i . \end{aligned}$$

We will use this type of manipulation frequently without explicit explanation.

It is customary in group theory to call the size of a group G its *order*. That is, the order of a group G is $|G|$, the number of elements in it. We will often make use of the following basic fact. It says that if any group element is raised to the power the order of the group, the result is the identity element of the group.

Fact 9.3 Let G be a group and let $m = |G|$ be its order. Then $a^m = \mathbf{1}$ for all $a \in G$. ▌

This means that computation in the group indices can be done modulo m:

Proposition 9.4 Let G be a group and let $m = |G|$ be its order. Then $a^i = a^{i \bmod m}$ for all $a \in G$ and all $i \in \mathbf{Z}$. ▌

We leave it to the reader to prove that this follows from Fact 9.3.

Example 9.5 Let us work in the group $\mathbf{Z}_{21}^*$ under the operation of multiplication modulo 21. The members of this group are $1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20$, so the order of the group is $m = 12$. Suppose we want to compute 5^{86} in this group. Applying the above we have

$$5^{86} \bmod 21 \;=\; 5^{86 \bmod 12} \bmod 21 \;=\; 5^2 \bmod 21 \;=\; 25 \bmod 21 \;=\; 4\,. \;▌$$

If G is a group, a set $S \subseteq G$ is called a subgroup if it is a group in its own right, under the same operation as that under which G is a group. If we already know that G is a group, there is a simple way to test whether S is a subgroup: it is one if and only if $x \cdot y^{-1} \in S$ for all $x, y \in S$. Here y^{-1} is the inverse of y in G.

Fact 9.6 Let G be a group and let S be a subgroup of G. Then the order of S divides the order of G. ▌

9.2 Algorithms

Fig. 9.1 summarizes some basic algorithms involving numbers. These algorithms are used to implement public-key cryptosystems, and thus their running time is an important concern. We begin with a discussion about the manner in which running time is measured, and then go on to discuss the algorithms, some very briefly, some in more depth.

9.2.1 Bit operations and binary length

In a course or text on algorithms, we learn to analyze the running time of an algorithm as a function of the size of its input. The inputs are typically things like graphs, or arrays, and the measure of input size might be the number of nodes in the graph or the length of the array. Within the algorithm we often need to perform arithmetic operations, like addition or multiplication of array indices. We typically assume these have $O(1)$ cost. The reason this assumption is reasonable is that the numbers in question are small and the cost of manipulating them is negligible compared to costs proportional to the size of the array or graph on which we are working.

In contrast, the numbers arising in cryptographic algorithms are large, having magnitudes like 2^{512} or 2^{1024}. The arithmetic operations on these numbers are the main cost of the algorithm, and the costs grow as the numbers get bigger.

The numbers are provided to the algorithm in binary, and the size of the input number is thus the number of bits in its binary representation. We call this the length, or binary length, of the number, and we measure the running time of the algorithm as a function of the binary lengths of its input numbers. In computing the running time, we count the number of bit operations performed.

Let $b_{k-1} \ldots b_1 b_0$ be the binary representation of a positive integer a, meaning $b_0, \ldots, b_{k-1}$ are bits such that $b_{k-1} = 1$ and $a = 2^{k-1} b_{k-1} + 2^{k-2} b_{k-2} + \cdots + 2^1 b_1 + 2^0 b_0$. Then the binary length of a is k, and is denoted $|a|$. Notice that $|a| = k$ if and only if $2^{k-1} \le a < 2^k$. If a is negative, we let $|a| = |-a|$, and assume that an additional bit or two is used to indicate to the algorithm that the input is negative.

Algorithm	Input	Output	Running Time
INT-DIV	$a, N \quad (N > 0)$	(q, r) with $a = Nq + r$ and $0 \leq r < N$	$O(\lvert a\rvert \cdot \lvert N\rvert)$
MOD	$a, N \quad (N > 0)$	$a \bmod N$	$O(\lvert a\rvert \cdot \lvert N\rvert)$
EXT-GCD	$a, b \quad ((a, b) \neq (0, 0))$	$(d, \overline{a}, \overline{b})$ with $d = \gcd(a, b) = a\overline{a} + b\overline{b}$	$O(\lvert a\rvert \cdot \lvert b\rvert)$
MOD-ADD	$a, b, N \quad (a, b \in \mathbf{Z}_N)$	$(a + b) \bmod N$	$O(\lvert N\rvert)$
MOD-MULT	$a, b, N \quad (a, b \in \mathbf{Z}_N)$	$ab \bmod N$	$O(\lvert N\rvert^2)$
MOD-INV	$a, N \quad (a \in \mathbf{Z}_N^*)$	$b \in \mathbf{Z}_N^*$ with $ab \equiv 1 \pmod N$	$O(\lvert N\rvert^2)$
MOD-EXP	$a, n, N \quad (a \in \mathbf{Z}_N)$	$a^n \bmod N$	$O(\lvert n\rvert \cdot \lvert N\rvert^2)$
EXP_G	$a, n \quad (a \in G)$	$a^n \in G$	$2\lvert n\rvert$ G-operations

Figure 9.1: **Some basic algorithms and their running time.** Unless otherwise indicated, an input value is an integer and the running time is the number of bit operations. G denotes a group.

9.2.2 Integer division and mod algorithms

We define the integer division function as taking input two integers a, N, with $N > 0$, and returning the quotient and remainder obtained by dividing a by N. That is, the function returns (q, r) such

that $a = qN + r$ with $0 \leq r < N$. We denote by INT-DIV an algorithm implementing this function. The algorithm uses the standard division method we learned way back in school, which turns out to run in time proportional to the product of the binary lengths of a and N.

We also want an algorithm that implements the mod function, taking integer inputs a, N with $N > 0$ and returning $a \bmod N$. This algorithm, denoted MOD, can be implemented simply by calling INT-DIV(a, N) to get (q, r), and then returning just the remainder r.

9.2.3 Extended GCD algorithm

Suppose a, b are integers, not both 0. A basic fact about the greatest common divisor of a and b is that it is the smallest positive element of the set

$$\{\, a\overline{a} + b\overline{b} \;:\; \overline{a}, \overline{b} \in \mathbf{Z} \,\}$$

of all integer linear combinations of a and b. In particular, if $d = \gcd(a, b)$ then there exist integers $\overline{a}, \overline{b}$ such that $d = a\overline{a} + b\overline{b}$. (Note that either $\overline{a}$ or $\overline{b}$ could be negative.)

Example 9.7 The gcd of 20 and 12 is $d = \gcd(20, 12) = 4$. We note that $4 = 20(2) + (12)(-3)$, so in this case $\overline{a} = 2$ and $\overline{b} = -3$. ▌

Besides the gcd itself, we will find it useful to be able to compute these weights $\overline{a}, \overline{b}$. This is what the extended-gcd algorithm EXT-GCD does: given a, b as input, it returns $(d, \overline{a}, \overline{b})$ such that $d = \gcd(a, b) = a\overline{a} + b\overline{b}$. The algorithm itself is an extension of Euclid's classic algorithm for computing the gcd, and the simplest description is a recursive one. We now provide it, and then discuss the correctness and running time. The algorithm takes input any integers a, b, not both zero.

```
Algorithm EXT-GCD(a, b)
If b = 0 then return (a, 1, 0)
Else
    (q, r) ← INT-DIV(a, b)
    (d, x, y) ← EXT-GCD(b, r)
    ā ← y
    b̄ ← x − qy
    Return (d, ā, b̄)
EndIf
```

The base case is when $b = 0$. If $b = 0$ then we know by assumption that $a \neq 0$, so $\gcd(a, b) = a$, and since $a = a(1) + b(0)$, the weights are 1 and 0. If $b \neq 0$ then we can divide by it, and we divide a by it to get a quotient q and remainder r. For the recursion, we use the fact that $\gcd(a, b) = \gcd(b, r)$. The recursive call thus yields $d = \gcd(a, b)$ together with weights x, y such that $d = bx + ry$. Noting that $a = bq + r$ we have

$$d \;=\; bx + ry \;=\; bx + (a - bq)y \;=\; ay + b(x - qy) \;=\; a\overline{a} + b\overline{b}\,,$$

confirming that the values assigned to $\overline{a}, \overline{b}$ are correct.

The running time of this algorithm is $O(|a| \cdot |b|)$, or, put a little more simply, the running time is quadratic in the length of the longer number. This is not so obvious, and proving it takes some work. We do not provide this proof here.

We also want to know an upper bound on the lengths of the weights $\overline{a}, \overline{b}$ output by EXT-GCD(a, b). The running time bound tells us that $|\overline{a}|, |\overline{b}| = O(|a| \cdot |b|)$, but this is not good enough for some of what follows. I would expect that $|\overline{a}|, |\overline{b}| = O(|a| + |b|)$. Is this true? If so, can it be proved by induction based on the recursive algorithm above?

9.2.4 Algorithms for modular addition and multiplication

The next two algorithms in Fig. 9.1 are the ones for modular addition and multiplication. To compute $(a + b) \bmod N$, we first compute $c = a + b$ using the usual algorithm we learned way back in school, which runs in time linear in the binary representations of the numbers. We might imagine that it now takes quadratic time to do the mod operation, but in fact if $c > N$, the mod operation can be simply executed by subtracting N from c, which takes only linear time, which is why the algorithm as a whole takes linear time. For multiplication mod N, the process is much the same. First compute $c = ab$ using the usual algorithm, which is quadratic time. This time we do the mod by invoking MOD(c, N). (The length of c is the sum of the lengths of a and b, and so c is not small as in the addition case, so a shortcut to the mod as we saw there does not seem possible.)

9.2.5 Algorithm for modular inverse

The next algorithm in Fig. 9.1 is for computation of the multiplicative inverse of a in the group $\mathbf{Z}_N^*$. Namely, on input $N > 0$ and $a \in \mathbf{Z}_N^*$, algorithm MOD-INV returns b such that $ab \equiv 1 \pmod{N}$. The method is quite simple:

Algorithm MOD-INV(a, N)
$(d, \overline{a}, \overline{N}) \leftarrow$ EXT-GCD(a, N)
$b \leftarrow \overline{a} \bmod N$
Return b

Correctness is easy to see. Since $a \in \mathbf{Z}_N^*$ we know that $\gcd(a, N) = 1$. The EXT-GCD algorithm thus guarantees that $d = 1$ and $1 = a\overline{a} + N\overline{N}$. Since $N \bmod N = 0$, we have $1 \equiv a\overline{a} \pmod{N}$, and thus $b = \overline{a} \bmod N$ is the right value to return.

The cost of the first step is $O(|a| \cdot |N|)$. The cost of the second step is $O(|\overline{a}| \cdot |N|)$. If we assume that $|\overline{a}| = O(|a| + |N|)$ then the overall cost is $O(|a| \cdot |N|)$. See discussion of the EXT-GCD algorithm regarding this assumption on the length of $\overline{a}$.

9.2.6 Exponentiation algorithm

We will be using exponentiation in various different groups, so it is useful to look at it at the group level. Let G be a group and let $a \in G$. Given an integer $n \in \mathbf{Z}$ we want to compute the group element a^n as defined in Section 9.1.2. The naive method, assuming for simplicity $n \geq 0$, is to execute

$y \leftarrow \mathbf{1}$
For $i = 1, \ldots, n$ do $y \leftarrow y \cdot a$ EndFor
Return y

This might at first seem like a satisfactory algorithm, but actually it is very slow. The number of group operations required is n, and the latter can be as large as the order of the group. Since we are often looking at groups containing about 2^{512} elements, exponentiation by this method is not feasible. In the language of complexity theory, the problem is that we are looking at an exponential time algorithm. This is because the running time is exponential in the binary length $|n|$ of the input n. So we seek a better algorithm. We illustrate the idea of fast exponentiation with an example.

Example 9.8 Suppose the binary length of n is 5, meaning the binary representation of n has the form $b_4b_3b_2b_1b_0$. Then

$$\begin{aligned} n &= 2^4b_4 + 2^3b_3 + 2^2b_2 + 2^1b_1 + 2^0b_0 \\ &= 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0 \ . \end{aligned}$$

Our exponentiation algorithm will proceed to compute the values $y_5, y_4, y_3, y_2, y_1, y_0$ in turn, as follows:

$$\begin{aligned} y_5 &= \mathbf{1} \\ y_4 &= y_5^2 \cdot a^{b_4} &&= a^{b_4} \\ y_3 &= y_4^2 \cdot a^{b_3} &&= a^{2b_4+b_3} \\ y_2 &= y_3^2 \cdot a^{b_2} &&= a^{4b_4+2b_3+b_2} \\ y_1 &= y_2^2 \cdot a^{b_1} &&= a^{8b_4+4b_3+2b_2+b_1} \\ y_0 &= y_1^2 \cdot a^{b_0} &&= a^{16b_4+8b_3+4b_2+2b_1+b_0} \ . \end{aligned}$$

Two group operations are required to compute y_i from y_{i+1}, and the number of steps equals the binary length of n, so the algorithm is fast. ▌

In general, we let $b_{k-1} \ldots b_1 b_0$ be the binary representation of n, meaning $b_0, \ldots, b_{k-1}$ are bits such that $n = 2^{k-1}b_{k-1} + 2^{k-2}b_{k-2} + \cdots + 2^1 b_1 + 2^0 b_0$. The algorithm proceeds as follows given any input $a \in G$ and $n \in \mathbf{Z}$:

Algorithm $\mathrm{EXP}_G(a, n)$
If $n < 0$ then $a \leftarrow a^{-1}$ and $n \leftarrow -n$ EndIf
Let $b_{k-1} \ldots b_1 b_0$ be the binary representation of n
$y \leftarrow \mathbf{1}$
For $i = k - 1$ downto 0 do
  $y \leftarrow y^2 \cdot a^{b_i}$
End For
Output y

The algorithm uses two group operations per iteration of the loop: one to multiply y by itself, another to multiply the result by a^{b_i}. (The computation of a^{b_i} is without cost, since this is just a if $b_i = 1$ and $\mathbf{1}$ if $b_i = 0$.) So its total cost is $2k = 2|n|$ group operations. (We are ignoring the cost of the one possible inversion in the case $n < 0$.) (This is the worst case cost. We observe that it actually takes $|n| + W_H(n)$ group operations, where $W_H(n)$ is the number of ones in the binary representation of n.)

We will typically use this algorithm when the group G is $\mathbf{Z}_N^*$ and the group operation is multiplication modulo N, for some positive integer N. We have denoted this algorithm by MOD-EXP in

Fig. 9.1. (The input a is not required to be relatively prime to N even though it usually will be, so is listed as coming from $\mathbf{Z}_N$.) In that case, each group operation is implemented via MOD-MULT and takes $O(|N|^2)$ time, so the running time of the algorithm is $O(|n| \cdot |N|^2)$. Since n is usually in $\mathbf{Z}_N$, this comes to $O(|N|^3)$. The salient fact to remember is that modular exponentiation is a cubic time algorithm.

9.3 Cyclic groups and generators

Let G be a group, let $\mathbf{1}$ denote its identity element, and let $m = |G|$ be the order of G. If $g \in G$ is any member of the group, the *order* of g is defined to be the least positive integer n such that $g^n = \mathbf{1}$. We let

$$\langle g \rangle \;=\; \{\, g^i \;:\; i \in \mathbf{Z}_n \,\} \;=\; \{g^0, g^1, \ldots, g^{n-1}\}$$

denote the set of group elements generated by g. A fact we do not prove, but is easy to verify, is that this set is a subgroup of G. The order of this subgroup (which, by definition, is its size) is just the order of g. Fact 9.6 tells us that the order n of g divides the order m of the group. An element g of the group is called a *generator* of G if $\langle g \rangle = G$, or, equivalently, if its order is m. If g is a generator of G then for every $a \in G$ there is a unique integer $i \in \mathbf{Z}_m$ such that $g^i = a$. This i is called the discrete logarithm of a to base g, and we denote it by $\mathrm{DLog}_{G,g}(a)$. Thus, $\mathrm{DLog}_{G,g}(\cdot)$ is a function that maps G to $\mathbf{Z}_m$, and moreover this function is a bijection, meaning one-to-one and onto. The function of $\mathbf{Z}_m$ to G defined by $i \mapsto g^i$ is called the discrete exponentiation function, and the discrete logarithm function is the inverse of the discrete exponentiation function.

Example 9.9 Let $p = 11$, which is prime. Then $Z_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ has order $p - 1 = 10$. Let us find the subgroups generated by group elements 2 and 5. We raise them to the powers $i = 0, \ldots, 9$. We get:

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6
$5^i \bmod 11$	1	5	3	4	9	1	5	3	4	9

Looking at which elements appear in the row corresponding to 2 and 5, respectively, we can determine the subgroups these group elements generate:

$$\begin{aligned} \langle 2 \rangle &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \\ \langle 5 \rangle &= \{1, 3, 4, 5, 9\} \,. \end{aligned}$$

Since $\langle 2 \rangle$ equals $\mathbf{Z}_{11}^*$, the element 2 is a generator. Since a generator exists, $\mathbf{Z}_{11}^*$ is cyclic. On the other hand, $\langle 5 \rangle \neq \mathbf{Z}_{11}^*$, so 5 is not a generator. The order of 2 is 10, while the order of 5 is 5. Note that these orders divide the order 10 of the group. The table also enables us to determine the discrete logarithms to base 2 of the different group elements:

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{\mathbf{Z}_{11}^*,2}(a)$	0	1	8	2	4	9	7	3	6	5

Later we will see a way of identifying all the generators given that we know one of them. ▌

The discrete exponentiation function is conjectured to be one-way (meaning the discrete logarithm function is hard to compute) for some cyclic groups G. Due to this fact we often seek cyclic

groups for cryptographic usage. Here are three sources of such groups. We will not prove any of the facts below; their proofs can be found in books on algebra.

Fact 9.10 Let p be a prime. Then the group $\mathbf{Z}_p^*$ is cyclic. ▌

The operation here is multiplication modulo p, and the size of this group is $\varphi(p) = p - 1$. This is the most common choice of group in cryptography.

Fact 9.11 Let G be a group and let $m = |G|$ be its order. If m is a prime number, then G is cyclic. ▌

In other words, any group having a prime number of elements is cyclic. Note that it is not for this reason that Fact 9.10 is true, since the order of $\mathbf{Z}_p^*$ (where p is prime) is $p-1$, which is even if $p \geq 3$ and 1 if $p = 2$, and is thus never a prime number.

The following is worth knowing if you have some acquaintance with finite fields. Recall that a field is a set F equipped with two operations, an addition and a multiplication. The identity element of the addition is denoted 0. When this is removed from the field, what remains is a group under multiplication. This group is always cyclic.

Fact 9.12 Let F be a finite field, and let $F^* = F - \{0\}$. Then F^* is a cyclic group under the multiplication operation of F. ▌

A finite field of order m exists if and only if $m = p^n$ for some prime p and integer $n \geq 1$. The finite field of order p is exactly $\mathbf{Z}_p$, so the case $n = 1$ of Fact 9.12 implies Fact 9.10. Another interesting special case of Fact 9.12 is when the order of the field is 2^n, meaning $p = 2$, yielding a cyclic group of order $2^n - 1$.

When we want to use a cyclic group G in cryptography, we will often want to find a generator for it. The process used is to pick group elements in some appropriate way, and then test each chosen element to see whether it is a generator. One thus has to solve two problems. One is how to test whether a given group element is a generator, and the other is what process to use to choose the candidate generators to be tested.

Let $m = |G|$ and let $\mathbf{1}$ be the identity element of G. The obvious way to test whether a given $g \in G$ is a generator is to compute the values $g^1, g^2, g^3, \ldots$, stopping at the first j such that $g^j = \mathbf{1}$. If $j = m$ then g is a generator. This test however can require up to m group operations, which is not efficient, given that the groups of interest are large, so we need better tests.

The obvious way to choose candidate generators is to cycle through the entire group in some way, testing each element in turn. Even with a fast test, this can take a long time, since the group is large. So we would also like better ways of picking candidates.

We address these problems in turn. Let us first look at testing whether a given $g \in G$ is a generator. One sees quickly that computing all powers of g as in $g^1, g^2, g^3, \ldots$ is not necessary. For example if we computed g^8 and found that this is not $\mathbf{1}$, then we know that $g^4 \neq \mathbf{1}$ and $g^2 \neq \mathbf{1}$ and $g \neq \mathbf{1}$. More generally, if we know that $g^j \neq \mathbf{1}$ then we know that $g^i \neq \mathbf{1}$ for all i dividing j. This tells us that it is better to first compute high powers of g, and use that to cut down the space of exponents that need further testing. The following Proposition pinpoints the optimal way to do this. It identifies a set of exponents $m_1, \ldots, m_n$ such that one need only test whether $g^{m_i} \neq \mathbf{1}$ for $i = 1, \ldots, n$. As we will argue later, this set is quite small.

Proposition 9.13 Let G be a cyclic group and let $m = |G|$ be the size of G. Let $p_1^{\alpha_1} \cdots p_n^{\alpha_n}$ be the prime factorization of m and let $m_i = m/p_i$ for $i = 1, \ldots, n$. Let $g \in G$. Then g is a generator of G if and only if

$$\text{For all } i = 1, \ldots, n: \quad g^{m_i} \neq \mathbf{1} \,, \tag{9.1}$$

where $\mathbf{1}$ is the identity element of G. ▌

Proof of Proposition 9.13: First suppose that g is a generator of G. Then we know that the smallest positive integer j such that $g^j = \mathbf{1}$ is $j = m$. Since $0 < m_i < m$, it must be that $g^{m_i} \neq \mathbf{1}$ for all $i = 1, \ldots, m$.

Conversely, suppose g satisfies the condition of Equation (9.1). We want to show that g is a generator. Let j be the order of g, meaning the smallest positive integer such that $g^j = \mathbf{1}$. Then we know that j must divide the order m of the group, meaning $m = dj$ for some integer $d \geq 1$. This implies that $j = p_1^{\beta_1} \cdots p_n^{\beta_n}$ for some integers $\beta_1, \ldots, \beta_n$ satisfying $0 \leq \beta_i \leq \alpha_i$ for all $i = 1, \ldots, n$. If $j < m$ then there must be some i such that $\beta_i < \alpha_i$, and in that case j divides m_i, which in turn implies $g^{m_i} = \mathbf{1}$ (because $g^j = \mathbf{1}$). So the assumption that Equation (9.1) is true implies that j cannot be strictly less than m, so the only possibility is $j = m$, meaning g is a generator. ▌

The number n of terms in the prime factorization of m cannot be more than $\lg(m)$, the binary logarithm of m. (This is because $p_i \geq 2$ and $\alpha_i \geq 1$ for all $i = 1, \ldots, n$.) So, for example, if the group has size about 2^{512}, then at most 512 tests are needed. So testing is quite efficient. One should note however that it requires knowing the prime factorization of m.

Let us now consider the second problem we discussed above, namely how to choose candidate group elements for testing. There seems little reason to think that trying all group elements in turn will yield a generator in a reasonable amount of time. Instead, we consider picking group elements at random, and then testing them. The probability of success in any trial is $|\mathsf{Gen}(G)|/|G|$. So the expected number of trials before we find a generator is $|G|/|\mathsf{Gen}(G)|$. To estimate the efficacy of this method, we thus need to know the number of generators in the group. The following Proposition gives a characterization of the generator set which in turn tells us its size.

Proposition 9.14 Let G be a cyclic group of order m, and let g be a generator of G. Then $\mathsf{Gen}(G) = \{\, g^i \in G \,:\, i \in \mathbf{Z}_m^* \,\}$ and $|\mathsf{Gen}(G)| = \varphi(m)$. ▌

That is, having fixed one generator g, a group element h is a generator if and only if its discrete logarithm to base g is relatively prime to the order m of the group. As a consequence, the number of generators is the number of integers in the range $1, \ldots, m-1$ that are relatively prime to m.

Proof of Proposition 9.14: Given that $\mathsf{Gen}(G) = \{\, g^i \in G \,:\, i \in \mathbf{Z}_m^* \,\}$, the claim about its size follows easily:

$$|\mathsf{Gen}(G)| \;=\; \left|\{\, g^i \in G \,:\, i \in \mathbf{Z}_m^* \,\}\right| \;=\; |\mathbf{Z}_m^*| \;=\; \varphi(m) \,.$$

We now prove that $\mathsf{Gen}(G) = \{\, g^i \in G \,:\, i \in \mathbf{Z}_m^* \,\}$. First, we show that if $i \in \mathbf{Z}_m^*$ then $g^i \in \mathsf{Gen}(G)$. Second, we show that if $i \in \mathbf{Z}_m - \mathbf{Z}_m^*$ then $g^i \notin \mathsf{Gen}(G)$.

So first suppose $i \in \mathbf{Z}_m^*$, and let $h = g^i$. We want to show that h is a generator of G. It suffices to show that the only possible value of $j \in \mathbf{Z}_m$ such that $h^j = \mathbf{1}$ is $j = 0$, so let us now show this. Let $j \in \mathbf{Z}_m$ be such that $h^j = \mathbf{1}$. Since $h = g^i$ we have

$$\mathbf{1} \;=\; h^j \;=\; g^{ij \bmod m} \,.$$

Since g is a generator, it must be that $ij \equiv 0 \pmod{m}$, meaning m divides ij. But $i \in \mathbf{Z}_m^*$ so $\gcd(i, m) = 1$. So it must be that m divides j. But $j \in \mathbf{Z}_m$ and the only member of this set divisible by m is 0, so $j = 0$ as desired.

Next, suppose $i \in \mathbf{Z}_m - \mathbf{Z}_m^*$ and let $h = g^i$. To show that h is not a generator it suffices to show that there is some non-zero $j \in \mathbf{Z}_m$ such that $h^j = \mathbf{1}$. Let $d = \gcd(i, m)$. Our assumption $i \in \mathbf{Z}_m - \mathbf{Z}_m^*$ implies that $d > 1$. Let $j = m/d$, which is a non-zero integer in $\mathbf{Z}_m$ because $d > 1$. Then the following shows that $h^j = \mathbf{1}$, completing the proof:

$$h^j \;=\; g^{ij} \;=\; g^{i \cdot m/d} \;=\; g^{m \cdot i/d} \;=\; (g^m)^{i/d} \;=\; \mathbf{1}^{i/d} \;=\; \mathbf{1}.$$

We used here the fact that d divides i and that $g^m = \mathbf{1}$. ▌

Example 9.15 Let us determine all the generators of the group $\mathbf{Z}_{11}^*$. Let us first use Proposition 9.13. The size of $\mathbf{Z}_{11}^*$ is $m = \varphi(11) = 10$, and the prime factorization of 10 is $2^1 \cdot 5^1$. Thus, the test for whether a given $a \in \mathbf{Z}_{11}^*$ is a generator is that $a^2 \not\equiv 1 \pmod{11}$ and $a^5 \not\equiv 1 \pmod{11}$. Let us compute $a^2 \bmod 11$ and $a^5 \bmod 11$ for all group elements a. We get:

a	1	2	3	4	5	6	7	8	9	10
$a^2 \bmod 11$	1	4	9	5	3	3	5	9	4	1
$a^5 \bmod 11$	1	10	1	1	1	10	10	10	1	10

The generators are those a for which the corresponding column has no entry equal to 1, meaning in both rows, the entry for this column is different from 1. So

$$\mathsf{Gen}(\mathbf{Z}_{11}^*) \;=\; \{2, 6, 7, 8\}\ .$$

Now, let us use Proposition 9.14 and double-check that we get the same thing. We saw in Example 9.9 that 2 was a generator of $\mathbf{Z}_{11}^*$. As per Proposition 9.14, the set of generators is

$$\mathsf{Gen}(\mathbf{Z}_{11}^*) \;=\; \{\ 2^i \bmod 11 \ :\ i \in \mathbf{Z}_{10}^*\ \}\ .$$

This is because the size of the group is $m = 10$. Now, $\mathbf{Z}_{10}^* = \{1, 3, 7, 9\}$. The values of $2^i \bmod 11$ as i ranges over this set can be obtained from the table in Example 9.9 where we computed all the powers of 2. So

$$\begin{aligned} \{\ 2^i \bmod 11 \ :\ i \in \mathbf{Z}_{10}^*\ \} &= \{2^1 \bmod 11, 2^3 \bmod 11, 2^7 \bmod 11, 2^9 \bmod 11\} \\ &= \{2, 6, 7, 8\}\ . \end{aligned}$$

This is the same set we obtained above via Proposition 9.13. If we try to find a generator by picking group elements at random and then testing using Proposition 9.13, each trial has probability of success $\varphi(10)/10 = 4/10$, so we would expect to find a generator in $10/4$ trials. We can optimize slightly by noting that 1 and -1 can never be generators, and thus we only need pick candidates randomly from $\mathbf{Z}_{11}^* - \{1, 10\}$. In that case, each trial has probability of success $\varphi(10)/8 = 4/8 = 1/2$, so we would expect to find a generator in 2 trials. ▌

When we want to work in a cyclic group in cryptography, the most common choice is to work over $\mathbf{Z}_p^*$ for a suitable prime p. The algorithm for finding a generator would be to repeat the process of picking a random group element and testing it, halting when a generator is found. In order to make this possible we choose p in such a way that the prime factorization of the order $p - 1$ of

$\mathbf{Z}_p^*$ is known. In order to make the testing fast, we choose p so that $p-1$ has few prime factors. Accordingly, it is common to choose p to equal $2q+1$ for some prime q. In this case, the prime factorization of $p-1$ is $2^1 q^1$, so we need raise a candidate to only two powers to test whether or not it is a generator. In choosing candidates, we optimize slightly by noting that 1 and -1 are never generators, and accordingly pick the candidates from $\mathbf{Z}_p^* - \{1, p-1\}$ rather than from $\mathbf{Z}_p^*$. So the algorithm is as follows:

Algorithm FIND-GEN(p)
$q \leftarrow (p-1)/2$
$\mathsf{found} \leftarrow 0$
While ($\mathsf{found} \neq 1$) do
  $g \stackrel{\$}{\leftarrow} \mathbf{Z}_p^* - \{1, p-1\}$
  If ($g^2 \bmod p \neq 1$) and ($g^q \bmod p \neq 1$) then $\mathsf{found} \leftarrow 1$
EndWhile
Return g

Proposition 9.13 tells us that the group element g returned by this algorithm is always a generator of $\mathbf{Z}_p^*$. By Proposition 9.14, the probability that an iteration of the algorithm is successful in finding a generator is

$$\frac{|\mathsf{Gen}(\mathbf{Z}_p^*)|}{|\mathbf{Z}_p^*| - 2} = \frac{\varphi(p-1)}{p-3} = \frac{\varphi(2q)}{2q-2} = \frac{q-1}{2q-2} = \frac{1}{2}\,.$$

Thus the expected number of iterations of the while loop is 2. Above, we used that fact that $\varphi(2q) = q-1$ which is true because q is prime.

9.4 Squares and non-squares

An element a of a group G is called a *square*, or *quadratic residue* if it has a square root, meaning there is some $b \in G$ such that $b^2 = a$ in G. We let

$$\mathsf{QR}(G) = \{\, g \in G \,:\, g \text{ is quadratic residue in } G \,\}$$

denote the set of all squares in the group G. We leave to the reader to check that this set is a subgroup of G.

We are mostly interested in the case where the group G is $\mathbf{Z}_N^*$ for some integer N. An integer a is called a *square mod N* or *quadratic residue mod N* if $a \bmod N$ is a member of $\mathsf{QR}(\mathbf{Z}_N^*)$. If $b^2 \equiv a \pmod N$ then b is called a square-root of $a \bmod N$. An integer a is called a *non-square mod N* or *quadratic non-residue mod N* if $a \bmod N$ is a member of $\mathbf{Z}_N^* - \mathsf{QR}(\mathbf{Z}_N^*)$. We will begin by looking at the case where $N = p$ is a prime. In this case we define a function $J_p\colon\ \mathbf{Z} \to \{-1, 1\}$ by

$$J_p(a) = \begin{cases} 1 & \text{if } a \text{ is a square mod } p \\ 0 & \text{if } a \bmod p = 0 \\ -1 & \text{otherwise.} \end{cases}$$

for all $a \in \mathbf{Z}$. We call $J_p(a)$ the *Legendre symbol* of a. Thus, the Legendre symbol is simply a compact notation for telling us whether or not its argument is a square modulo p.

Before we move to developing the theory, it may be useful to look at an example.

Example 9.16 Let $p = 11$, which is prime. Then $\mathbf{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ has order $p-1 = 10$. A simple way to determine $\mathsf{QR}(\mathbf{Z}_{11}^*)$ is to square all the group elements in turn:

a	1	2	3	4	5	6	7	8	9	10
$a^2 \bmod 11$	1	4	9	5	3	3	5	9	4	1

The squares are exactly those elements that appear in the second row, so

$$\mathsf{QR}(\mathbf{Z}_{11}^*) \;=\; \{1, 3, 4, 5, 9\}\ .$$

The number of squares is 5, which we notice equals $(p-1)/2$. This is not a coincidence, as we will see. Also notice that each square has exactly two different square roots. (The square roots of 1 are 1 and 10; the square roots of 3 are 5 and 6; the square roots of 4 are 2 and 9; the square roots of 5 are 4 and 7; the square roots of 9 are 3 and 8.)

Since 11 is prime, we know that $\mathbf{Z}_{11}^*$ is cyclic, and as we saw in Example 9.9, 2 is a generator. (As a side remark, we note that a generator must be a non-square. Indeed, if $a = b^2$ is a square, then $a^5 = b^{10} = 1$ modulo 11 because 10 is the order of the group. So $a^j = 1$ modulo 11 for some positive $j < 10$, which means a is not a generator. However, not all non-squares need be generators.) Below, we reproduce from that example the table of discrete logarithms of the group elements. We also add below it a row providing the Legendre symbols, which we know because, above, we identified the squares. We get:

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{\mathbf{Z}_{11}^*,2}(a)$	0	1	8	2	4	9	7	3	6	5
$J_{11}(a)$	1	-1	1	1	1	-1	-1	-1	1	-1

We observe that the Legendre symbol of a is 1 if its discrete logarithm is even, and -1 if the discrete logarithm is odd, meaning the squares are exactly those group elements whose discrete logarithm is even. It turns out that this fact is true regardless of the choice of generator. ▌

As we saw in the above example, the fact that $\mathbf{Z}_p^*$ is cyclic is useful in understanding the structure of the subgroup of quadratic residues $\mathsf{QR}(\mathbf{Z}_p^*)$. The following Proposition summarizes some important elements of this connection.

Proposition 9.17 Let $p \geq 3$ be a prime and let g be a generator of $\mathbf{Z}_p^*$. Then

$$\mathsf{QR}(\mathbf{Z}_p^*) \;=\; \{\, g^i \;:\; i \in \mathbf{Z}_{p-1} \text{ and } i \text{ is even} \,\}\ , \tag{9.2}$$

and the number of squares mod p is

$$\left|\mathsf{QR}(\mathbf{Z}_p^*)\right| \;=\; \frac{p-1}{2}\ .$$

Furthermore, every square mod p has exactly two different square roots mod p. ▌

Proof of Proposition 9.17: Let

$$E \;=\; \{\, g^i \;:\; i \in \mathbf{Z}_{p-1} \text{ and } i \text{ is even} \,\}\ .$$

We will prove that $E = \mathsf{QR}(\mathbf{Z}_p^*)$ by showing first that $E \subseteq \mathsf{QR}(\mathbf{Z}_p^*)$ and second that $\mathsf{QR}(\mathbf{Z}_p^*) \subseteq E$.

To show that $E \subseteq \mathsf{QR}(\mathbf{Z}_p^*)$, let $a \in E$. We will show that $a \in \mathsf{QR}(\mathbf{Z}_p^*)$. Let $i = \mathrm{DLog}_{\mathbf{Z}_p^*,g}(a)$. Since

$a \in E$ we know that i is even. Let $j = i/2$ and note that $j \in \mathbf{Z}_{p-1}$. Clearly

$$(g^j)^2 \equiv g^{2j \bmod p-1} \equiv g^{2j} \equiv g^i \pmod{p},$$

so g^j is a square root of $a = g^i$. So a is a square.

To show that $\mathsf{QR}(\mathbf{Z}_p^*) \subseteq E$, let b be any element of $\mathbf{Z}_p^*$. We will show that $b^2 \in E$. Let $j = \mathrm{DLog}_{\mathbf{Z}_p^*,g}(b)$. Then

$$b^2 \equiv (g^j)^2 \equiv g^{2j \bmod p-1} \equiv g^{2j} \pmod{p},$$

the last equivalence being true because the order of the group $\mathbf{Z}_p^*$ is $p-1$. This shows that $b^2 \in E$.

The number of even integers in $\mathbf{Z}_{p-1}$ is exactly $(p-1)/2$ since $p-1$ is even. The claim about the size of $\mathsf{QR}(\mathbf{Z}_p^*)$ thus follows from Equation (9.2). It remains to justify the claim that every square mod p has exactly two square roots mod p. This can be seen by a counting argument, as follows.

Suppose a is a square mod p. Let $i = \mathrm{DLog}_{\mathbf{Z}_p^*,g}(a)$. We know from the above that i is even. Let $x = i/2$ and let $y = x + (p-1)/2 \bmod (p-1)$. Then g^x is a square root of a. Furthermore

$$(g^y)^2 \equiv g^{2y} \equiv g^{2x+(p-1)} \equiv g^{2x} g^{p-1} \equiv a \cdot 1 \equiv a \pmod{p},$$

so g^y is also a square root of a. Since i is an even number in $\mathbf{Z}_{p-1}$ and $p-1$ is even, it must be that $0 \leq x < (p-1)/2$. It follows that $(p-1)/2 \leq y < p-1$. Thus $x \neq y$. This means that a has as least two square roots. This is true for each of the $(p-1)/2$ squares mod p. So the only possibility is that each of these squares has exactly two square roots. ∎

Suppose we are interested in knowing whether or not a given $a \in \mathbf{Z}_p^*$ is a square mod p, meaning we want to know the value of the Legendre symbol $J_p(a)$. Proposition 9.17 tells us that

$$J_p(a) \;=\; (-1)^{\mathrm{DLog}_{\mathbf{Z}_p^*,g}(a)},$$

where g is any generator of $\mathbf{Z}_p^*$. This however is not very useful in computing $J_p(a)$, because it requires knowing the discrete logarithm of a, which is hard to compute. The following Proposition says that the Legendre symbols of a modulo an odd prime p can be obtained by raising a to the power $(p-1)/2$, and helps us compute the Legendre symbol.

Proposition 9.18 Let $p \geq 3$ be a prime. Then

$$J_p(a) \equiv a^{\frac{p-1}{2}} \pmod{p}$$

for any $a \in \mathbf{Z}_p^*$. ∎

Now one can determine whether or not a is a square mod p by running the algorithm MOD-EXP on inputs $a, (p-1)/2, p$. If the algorithm returns 1 then a is a square mod p, and if it returns $p-1$ (which is the same as $-1 \bmod p$) then a is a non-square mod p. Thus, the Legendre symbol can be computed in time cubic in the length of p.

Towards the proof of Proposition 9.18, we begin with the following lemma which is often useful in its own right.

Lemma 9.19 Let $p \geq 3$ be a prime. Then

$$g^{\frac{p-1}{2}} \equiv -1 \pmod{p}$$

for any generator g of $\mathbf{Z}_p^*$.

Proof of Lemma 9.19: We begin by observing that 1 and -1 are both square roots of 1 mod p, and are distinct. (It is clear that squaring either of these yields 1, so they are square roots of 1. They are distinct because -1 equals $p-1$ mod p, and $p-1 \neq 1$ because $p \geq 3$.) By Proposition 9.17, these are the only square roots of 1. Now let

$$b \;=\; g^{\frac{p-1}{2}} \bmod p \;.$$

Then $b^2 \equiv 1 \pmod{p}$, so b is a square root of 1. By the above b can only be 1 or -1. However, since g is a generator, b cannot be 1. (The smallest positive value of i such that g^i is 1 mod p is $i = p-1$.) So the only choice is that $b \equiv -1 \pmod{p}$, as claimed. ▌

Proof of Proposition 9.18: By definition of the Legendre symbol, we need to show that

$$a^{\frac{p-1}{2}} \equiv \begin{cases} 1 \pmod{p} & \text{if } a \text{ is a square mod } p \\ -1 \pmod{p} & \text{otherwise.} \end{cases}$$

Let g be a generator of $\mathbf{Z}_p^*$ and let $i = \mathrm{DLog}_{\mathbf{Z}_p^*,g}(a)$. We consider separately the cases of a being a square and a being a non-square.

Suppose a is a square mod p. Then Proposition 9.17 tells us that i is even. In that case

$$a^{\frac{p-1}{2}} \equiv (g^i)^{\frac{p-1}{2}} \equiv g^{i\cdot\frac{p-1}{2}} \equiv (g^{p-1})^{i/2} \equiv 1 \pmod{p} \;,$$

as desired.

Now suppose a is a non-square mod p. Then Proposition 9.17 tells us that i is odd. In that case

$$a^{\frac{p-1}{2}} \equiv (g^i)^{\frac{p-1}{2}} \equiv g^{i\cdot\frac{p-1}{2}} \equiv g^{(i-1)\cdot\frac{p-1}{2}+\frac{p-1}{2}} \equiv (g^{p-1})^{(i-1)/2}\cdot g^{\frac{p-1}{2}} \equiv g^{\frac{p-1}{2}} \pmod{p} \;.$$

However Lemma 9.19 tells us that the last quantity is -1 modulo p, as desired. ▌

The following Proposition says that $ab \bmod p$ is a square if and only if either both a and b are squares, or if both are non-squares. But if one is a square and the other is not, then $ab \bmod p$ is a non-square. This can be proved by using either Proposition 9.17 or Proposition 9.18. We use the latter in the proof. You might try, as an exercise, to reprove the result using Proposition 9.17 instead.

Proposition 9.20 Let $p \geq 3$ be prime. Then

$$J_p(ab \bmod p) = J_p(a)\cdot J_p(b)$$

for all $a, b \in \mathbf{Z}_p^*$. ▌

Proof of Proposition 9.20: Using Proposition 9.18 we get

$$J_p(ab \bmod p) \equiv (ab)^{\frac{p-1}{2}} \equiv a^{\frac{p-1}{2}} b^{\frac{p-1}{2}} \equiv J_p(a)\cdot J_p(b) \pmod{p} \;.$$

The two quantities we are considering both being either 1 or -1, and equal modulo p, must then be actually equal. ▌

A quantity of cryptographic interest is the Diffie-Hellman (DH) key. Having fixed a cyclic group G and generator g for it, the DH key associated to elements $X = g^x$ and $Y = g^y$ of the group is the group element g^{xy}. The following Proposition tells us that the DH key is a square if either X *or* Y is a square, and otherwise is a non-square.

Proposition 9.21 Let $p \geq 3$ be a prime and let g be a generator of $\mathbf{Z}_p^*$. Then

$$J_p(g^{xy} \bmod p) = 1 \quad \text{if and only if} \quad J_p(g^x \bmod p) = 1 \text{ or } J_p(g^y \bmod p) = 1 \,,$$

for all $x, y \in \mathbf{Z}_{p-1}$. ∎

Proof of Proposition 9.21: By Proposition 9.17, it suffices to show that

$$xy \bmod (p-1) \text{ is even} \quad \text{if and only if} \quad x \text{ is even or } y \text{ is even} \,.$$

But since $p-1$ is even, $xy \bmod (p-1)$ is even exactly when xy is even, and clearly xy is even exactly if either x or y is even. ∎

With a cyclic group G and generator g of G fixed, we will be interested in the distribution of the DH key g^{xy} in G, under random choices of x, y from $\mathbf{Z}_m$, where $m = |G|$. One might at first think that in this case the DH key is a random group element. The following proposition tells us that in the group $\mathbf{Z}_p^*$ of integers modulo a prime, this is certainly not true. The DH key is significantly more likely to be a square than a non-square, and in particular is thus not even almost uniformly distributed over the group.

Proposition 9.22 Let $p \geq 3$ be a prime and let g be a generator of $\mathbf{Z}_p^*$. Then

$$\Pr\left[x \xleftarrow{\$} \mathbf{Z}_{p-1} \,;\, y \xleftarrow{\$} \mathbf{Z}_{p-1} \,:\, J_p(g^{xy}) = 1 \right]$$

equals $3/4$. ∎

Proof of Proposition 9.22: By Proposition 9.22 we need only show that

$$\Pr\left[x \xleftarrow{\$} \mathbf{Z}_{p-1} \,;\, y \xleftarrow{\$} \mathbf{Z}_{p-1} \,:\, J_p(g^x) = 1 \text{ or } J_p(g^y) = 1 \right]$$

equals $3/4$. The probability in question is $1 - \alpha$ where

$$\begin{aligned}
\alpha &= \Pr\left[x \xleftarrow{\$} \mathbf{Z}_{p-1} \,;\, y \xleftarrow{\$} \mathbf{Z}_{p-1} \,:\, J_p(g^x) = -1 \text{ and } J_p(g^y) = -1 \right] \\
&= \Pr\left[x \xleftarrow{\$} \mathbf{Z}_{p-1} \,:\, J_p(g^x) = -1 \right] \cdot \Pr\left[y \xleftarrow{\$} \mathbf{Z}_{p-1} \,:\, J_p(g^y) = -1 \right] \\
&= \frac{|\mathsf{QR}(\mathbf{Z}_p^*)|}{|\mathbf{Z}_p^*|} \cdot \frac{|\mathsf{QR}(\mathbf{Z}_p^*)|}{|\mathbf{Z}_p^*|} \\
&= \frac{(p-1)/2}{p-1} \cdot \frac{(p-1)/2}{p-1} \\
&= \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{1}{4} \,.
\end{aligned}$$

Thus $1 - \alpha = 3/4$ as desired. Here we used Proposition 9.17 which told us that $|\mathsf{QR}(\mathbf{Z}_p^*)| = (p-1)/2$. ∎

The above Propositions, combined with Proposition 9.18 (which tells us that quadratic residuosity modulo a prime can be efficiently tested), will later lead us to pinpoint weaknesses in certain cryptographic schemes in $\mathbf{Z}_p^*$.

9.5 Groups of prime order

A group of prime order is a group G whose order $m = |G|$ is a prime number. Such a group is always cyclic. These groups turn out to be quite useful in cryptography, so let us take a brief look at them and some of their properties.

An element h of a group G is called *non-trivial* if it is not equal to the identity element of the group.

Proposition 9.23 Suppose G is a group of order q where q is a prime, and h is any non-trivial member of G. Then h is a generator of G. ▌

Proof of Proposition 9.23: It suffices to show that the order of h is q. We know that the order of any group element must divide the order of the group. Since the group has prime order q, the only possible values for the order of h are 1 and q. But h does not have order 1 since it is non-trivial, so it must have order q. ▌

A common way to obtain a group of prime order for cryptographic schemes is as a subgroup of a group of integers modulo a prime. We pick a prime p having the property that $q = (p-1)/2$ is also prime. It turns out that the subgroup of quadratic residues modulo p then has order q, and hence is a group of prime order. The following proposition summarizes the facts for future reference.

Proposition 9.24 Let $q \geq 3$ be a prime such that $p = 2q + 1$ is also prime. Then $\mathsf{QR}(\mathbf{Z}_p^*)$ is a group of prime order q. Furthermore, if g is any generator of $\mathbf{Z}_p^*$, then $g^2 \bmod p$ is a generator of $\mathsf{QR}(\mathbf{Z}_p^*)$. ▌

Note that the operation under which $\mathsf{QR}(\mathbf{Z}_p^*)$ is a group is multiplication modulo p, the same operation under which $\mathbf{Z}_p^*$ is a group.

Proof of Proposition 9.24: We know that $\mathsf{QR}(\mathbf{Z}_p^*)$ is a subgroup, hence a group in its own right. Proposition 9.17 tells us that $|\mathsf{QR}(\mathbf{Z}_p^*)|$ is $(p-1)/2$, which equals q in this case. Now let g be a generator of $\mathbf{Z}_p^*$ and let $h = g^2 \bmod p$. We want to show that h is a generator of $\mathsf{QR}(\mathbf{Z}_p^*)$. As per Proposition 9.23, we need only show that h is non-trivial, meaning $h \neq 1$. Indeed, we know that $g^2 \not\equiv 1 \pmod{p}$, because g, being a generator, has order p and our assumptions imply $p > 2$. ▌

Example 9.25 Let $q = 5$ and $p = 2q + 1 = 11$. Both p and q are primes. We know from Example 9.16 that

$$\mathsf{QR}(\mathbf{Z}_{11}^*) = \{1, 3, 4, 5, 9\} .$$

This is a group of prime order 5. We know from Example 9.9 that 2 is a generator of $\mathbf{Z}_p^*$. Proposition 9.24 tells us that $4 = 2^2$ is a generator of $\mathsf{QR}(\mathbf{Z}_{11}^*)$. We can verify this by raising 4 to the powers $i = 0, \ldots, 4$:

i	0	1	2	3	4
$4^i \bmod 11$	1	4	5	9	3

We see that the elements of the last row are exactly those of the set $\mathsf{QR}(\mathbf{Z}_{11}^*)$. ▌

Let us now explain what we perceive to be the advantage conferred by working in a group of prime order. Let G be a cyclic group, and g a generator. We know that the discrete logarithms to

base g range in the set $\mathbf{Z}_m$ where $m = |G|$ is the order of G. This means that arithmetic in these exponents is modulo m. If G has prime order, then m is prime. This means that *any non-zero exponent has a multiplicative inverse modulo* m. In other words, in working in the exponents, we can divide. It is this that turns out to be useful.

As an example illustrating how we use this, let us return to the problem of the distribution of the DH key that we looked at in Section 9.4. Recall the question is that we draw x, y independently at random from $\mathbf{Z}_m$ and then ask how g^{xy} is distributed over G. We saw that when $G = \mathbf{Z}_p^*$ for a prime $p \geq 3$, this distribution was noticebly different from uniform. In a group of prime order, the distribution of the DH key, in contrast, is very close to uniform over G. It is not quite uniform, because the identity element of the group has a slightly higher probability of being the DH key than other group elements, but the deviation is small enough to be negligible for groups of reasonably large size. The following proposition summarizes the result.

Proposition 9.26 Suppose G is a group of order q where q is a prime, and let g be a generator of G. Then for any $Z \in G$ we have

$$\Pr\left[\, x \xleftarrow{\$} \mathbf{Z}_q \,;\, y \xleftarrow{\$} \mathbf{Z}_q \,:\, g^{xy} = Z \,\right] \;=\; \begin{cases} \dfrac{1}{q}\left(1 - \dfrac{1}{q}\right) & \text{if } Z \neq \mathbf{1} \\ \dfrac{1}{q}\left(2 - \dfrac{1}{q}\right) & \text{if } Z = \mathbf{1}, \end{cases}$$

where $\mathbf{1}$ denotes the identity element of G. ▌

Proof of Proposition 9.26: First suppose $Z = \mathbf{1}$. The DH key g^{xy} is $\mathbf{1}$ if and only if either x or y is 0 modulo q. Each is 0 with probability $1/q$ and these probabilities are independent, so the probability that either x or y is 0 is $2/q - 1/q^2$, as claimed.

Now suppose $Z \neq \mathbf{1}$. Let $z = \mathrm{DLog}_{G,g}(Z)$, meaning $z \in \mathbf{Z}_q^*$ and $g^z = Z$. We will have $g^{xy} \equiv Z \pmod{p}$ if and only if $xy \equiv z \pmod{q}$, by the uniqueness of the discrete logarithm. For any fixed $x \in \mathbf{Z}_q^*$, there is exactly one $y \in \mathbf{Z}_q$ for which $xy \equiv z \pmod{q}$, namely $y = x^{-1}z \bmod q$, where x^{-1} is the multiplicative inverse of x in the group $\mathbf{Z}_q^*$. (Here we are making use of the fact that q is prime, since otherwise the inverse of x modulo q may not exist.) Now, suppose we choose x at random from $\mathbf{Z}_q$. If $x = 0$ then, regardless of the choice of $y \in \mathbf{Z}_q$, we will not have $xy \equiv z \pmod{q}$, because $z \not\equiv 0 \pmod{q}$. On the other hand, if $x \neq 0$ then there is exactly $1/q$ probability that the randomly chosen y is such that $xy \equiv z \pmod{q}$. So the probability that $xy \equiv z \pmod{q}$ when both x and y are chosen at random in $\mathbf{Z}_q$ is

$$\frac{q-1}{q} \cdot \frac{1}{q} \;=\; \frac{1}{q}\left(1 - \frac{1}{q}\right)$$

as desired. Here, the first term is because when we choose x at random from $\mathbf{Z}_q$, it has probability $(q-1)/q$ of landing in $\mathbf{Z}_q^*$. ▌

9.6 Historical Notes

9.7 Exercises and Problems