2.1.3 The group axioms

All but one of the group axioms are now concrete. Namely, for *closure*, if we start with two points in $E(K)$, then the chord-and-tangent process gives rise to a cubic polynomial in K for which two roots (the two x-coordinates of the points we started with) are in K, meaning the third root must also be in K; the explicit formulas affirm this. The *identity* and *inverse* axioms are fine, since $P \oplus \mathcal{O} = P$, and the element $\ominus P$ such that $P \oplus (\ominus P) = \mathcal{O}$ is clearly unique and well defined for all P. We also note that the group is *abelian*, since the process of computing $P \oplus Q$ is symmetric. The only non-obvious axiom is *associativity*, i.e. showing $(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$. An elementary approach using the explicit formulas above can be used to show associativity by treating all the separate cases, but this approach is rather messy [Fri05]. Silverman gives a much more instructive proof [Sil09, Ch. III.3.4e] using tools that we will develop in the following chapter, but for now we offer some temporary intuition via the illustration in Figures 2.12 and 2.13.

2.1.4 Speeding up elliptic curve computations

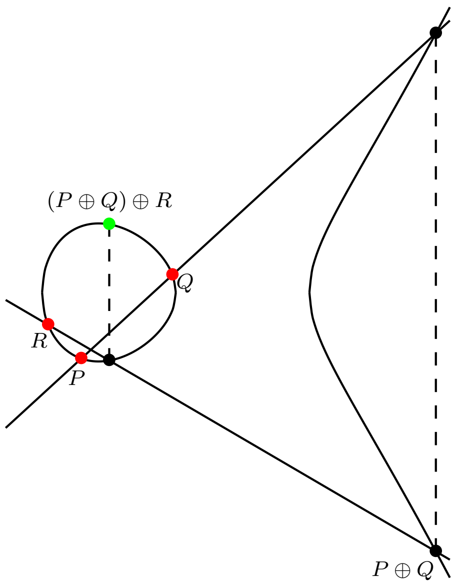

Figure 2.12: $(P \oplus Q) \oplus R$.

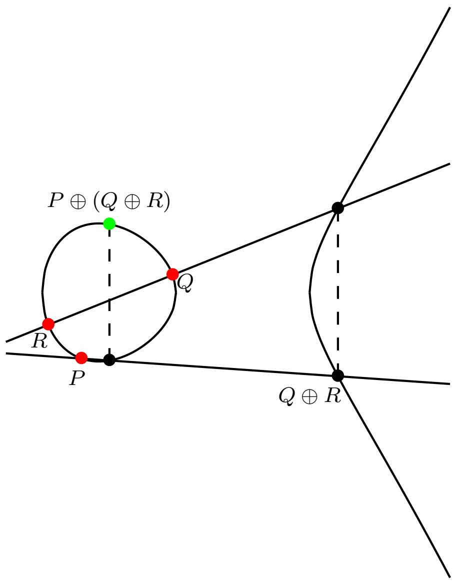

Figure 2.13: $P \oplus (Q \oplus R)$.

Group law computations on elliptic curves are clearly more complicated than computations in traditional groups that facilitate discrete logarithm based protocols like $\mathbb{F}_q^*$; the explicit formulas in (2.4) and (2.5) use many field operations.

However, in the context of cryptography, the more abstract nature of elliptic curve groups actually works in their favour. This is essentially because attackers aiming to solve the discrete logarithm problem on elliptic curves also face this abstractness. The subexponential algorithms that apply to finite field discrete logarithms[1] do not translate to the elliptic curve setting, where the best available attacks remain generic, exponential algorithms like Pollard rho [Pol78]. This means that elliptic curve groups of a relatively small size achieves the same conjectured security as multiplicative groups in much larger finite fields, i.e. $E(\mathbb{F}_{q_1})$ and $\mathbb{F}_{q_2}^*$ achieve similar security when $q_2 \gg q_1$. For example, an elliptic curve defined over a 160-bit field currently offers security comparable to a finite field of 1248 bits [Sma10, Table 7.2]. Thus, although more field operations are required to perform a group law computation, these operations take place in a field whose operational complexity is much less, and this difference is more than enough to tip the balance in the favour of elliptic curves. In addition, the smaller group elements in $E(\mathbb{F}_{q_1})$ implies much smaller key sizes, greatly reducing storage and bandwidth requirements. These are some of the major reasons that elliptic curves have received so much attention in the realm of public-key cryptography; the field of elliptic curve cryptography (ECC) has been thriving since Koblitz [Kob87] and Miller [Mil85] independently suggested their potential as alternatives to traditional groups.

One avenue of research that has given ECC a great boost is that of optimising the group law computations. The explicit formulas in affine coordinates ((2.4) and (2.5)) would not be used to compute the group law in practice, and in fact the Weierstrass model $E : y^2 = x^3 + ax + b$ is often not the optimal curve model either. A huge amount of effort has been put towards investigating other models and coordinate systems in order to minimise the field operations required in group law computations. One of the initial leaps forward in this line of research was the observation that performing computations in projective space avoids field inversions, which are extremely costly in practice. We illustrate these techniques in the following examples.

Example 2.1.9 (Magma script). Consider a general Weierstrass curve $E(\mathbb{F}_q) : y^2 = x^3 + ax + b$ where q is a large prime, and let $\mathbf{M}$, $\mathbf{S}$ and $\mathbf{I}$ represent the cost of computing multiplications, squarings and inversions in $\mathbb{F}_q$ respectively. To compute a general affine point doubling $(x_R, y_R) = [2](x_P, y_P)$ using (2.5) costs

[1]See Diem's notes on *index calculus* for a nice introduction [Die12].

$2\mathbf{M}+2\mathbf{S}+\mathbf{I}$, and to compute a general affine point addition $(x_R, y_R) = (x_P, y_P) \oplus (x_Q, y_Q)$ using (2.4) costs $2\mathbf{M}+\mathbf{S}+\mathbf{I}$. On the other hand, we can transform the formulas into homogeneous projective space according to the substitutions $x = X/Z$ and $y = Y/Z$, and we can consider computing $(X_R : Y_R : Z_R) = [2](X_P : Y_P : Z_P)$ and $(X_R : Y_R : Z_R) = (X_P : Y_P : Z_P) \oplus (X_Q : Y_Q : Z_Q)$ on $E : Y^2Z = X^3 + aXZ^2 + bZ^3$. For the addition case, substituting $x_i = X_i/Z_i$ and $y_i = Y_i/Z_i$ for $i \in \{P, Q, R\}$ into the affine formulas

$$x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q; \qquad y_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_R) - y_P$$

taken from (2.4), gives

$$\frac{X_R}{Z_R} = \left(\frac{\frac{Y_Q}{Z_Q} - \frac{Y_P}{Z_P}}{\frac{X_Q}{Z_Q} - \frac{X_P}{Z_P}}\right)^2 - \frac{X_P}{Z_P} - \frac{X_Q}{Z_Q}; \qquad \frac{Y_R}{Z_R} = \left(\frac{\frac{Y_Q}{Z_Q} - \frac{Y_P}{Z_P}}{\frac{X_Q}{Z_Q} - \frac{X_P}{Z_P}}\right)\left(\frac{X_P}{Z_P} - \frac{X_R}{Z_R}\right) - \frac{Y_P}{Z_P}.$$

After a little manipulation, we can then set Z_R to be the smallest value that contains both denominators above, and update the numerators accordingly to give

$$\begin{aligned}
X_R &= (X_PZ_Q - X_QZ_P)\left(Z_PZ_Q(Y_PZ_Q - Y_QZ_P)^2 - (X_PZ_Q - X_QZ_P)^2(X_PZ_Q + X_QZ_P)\right);\\
Y_R &= Z_PZ_Q(X_QY_P - X_PY_Q)(X_PZ_Q - X_QZ_P)^2\\
&\quad - (Y_PZ_Q - Y_QZ_P)\left((Y_PZ_Q - Y_QZ_P)^2Z_PZ_Q - (X_PZ_Q + X_QZ_P)(X_PZ_Q - X_QZ_P)^2\right);\\
Z_R &= Z_PZ_Q(X_PZ_Q - X_QZ_P)^3.
\end{aligned}$$

The explicit formulas database (EFD) [BL07a] reports that the above formulas can be computed in a total of $12\mathbf{M}+2\mathbf{S}$. The real power of adopting projective coordinates for computations becomes apparent when we remark that most optimised implementations of $\mathbb{F}_q$ arithmetic have $\mathbf{I} \gg 20\mathbf{M}$, and the multiplication to inversion ratio is commonly reported to be $80 : 1$ or higher. Thus, the $12\mathbf{M}+2\mathbf{S}$ used for additions in projective space will be much faster than the $2\mathbf{M}+\mathbf{S}+\mathbf{I}$ for affine additions. For completeness, we remark that deriving the projective formulas for computing $(X_R : Y_R : Z_R) = [2](X_P : Y_P : Z_P)$ is analogous (but substantially more compact since we only have the projective coordinates of P to deal with), and the EFD reports that this can be done in $5\mathbf{M}+6\mathbf{S}$, which will again be much faster than the $2\mathbf{M}+2\mathbf{S}+I$ in affine space.

The Weierstrass model for elliptic curves covers all isomorphism classes, meaning that every elliptic curve can be written in Weierstrass form. Other models of elliptic curves are usually available if some condition holds, and (if this is the case) it can be advantageous to adopt such a model, as the following example shows.

Example 2.1.10 (Magma script). If x^3+ax+b has a root in $\mathbb{F}_q$, then Billet and Joye [BJ03, Eq. 8-10] show that instead of working with $E : y^2 = x^3+ax+b$, we can work with the (birationally equivalent) *Jacobi-quartic* curve $J : v^2 = au^4+du^2+1$, for appropriately defined a, d (that depend on the root). Here we write J using (u, v) coordinates so back-and-forth mappings are defined without confusion. Thus, consider $E/\mathbb{F}_{97} : y^2 = x^3+5x+5$, for which x^3+5x+5 has 34 as a root, so we will work on the isomorphic curve $J/\mathbb{F}_{97} : v^2 = 73u^4+46u^2+1$. Instead of homogeneous projective coordinates, [BJ03] projectified under the substitution $u = U/W$ and $v = V/W^2$, which gives the (non-homogeneous) projective closure as $J : V^2 = 73U^4+46U^2W^2+W^4$. Any point $(x, y) \neq \mathcal{O}$ on E can be taken straight to the projective closure of J via

$$(x, y) \mapsto \left(2(x-34) : (2x+34)(x-34)^2 - y^2 : y\right),$$

with the reverse mapping given by

$$(U : V : W) \mapsto \left(2\frac{V+W^2}{U^2} - 17, W\frac{4(V+W^2)-5U^2}{U^3}\right).$$

For example $(x, y) = (77, 21)$ maps to $(U : V : W) = (86 : 8 : 21)$, and vice versa. We now look at the formulas for the point addition $(U_3 : V_3 : W_3) = (U_1 : V_1 : W_1) \oplus (U_2 : V_2 : W_2)$ on $J : V^2 = aU^4 + dU^2W^2 + W^4$, taken from [BJ03, Eq. 11], as

$$\begin{aligned}
U_3 &= U_1W_1V_2 + U_2W_2V_1,\\
V_3 &= \left((W_1W_2)^2 + a(U_1U_2)^2\right)(V_1V_2 + dU_1U_2W_1W_2) + 2aU_1U_2W_1W_2(U_1^2W_2^2 + U_2^2W_1^2),\\
W_3 &= (W_1W_2)^2 - a(U_1U_2)^2,
\end{aligned}$$

where we immediately highlight the relative simplicity of the above formulas in comparison to the homogeneous projective formulas derived in the previous example. Unsurprisingly then, the fastest formulas for Jacobi-quartic additions and doublings outdo those for general Weierstrass curves in homogeneous projective space. Namely, the current fastest formulas for doublings on Jacobi-quartics cost $2\mathbf{M} + 5\mathbf{S}$ and additions cost $6\mathbf{M} + 4\mathbf{S}$ [HWCD09], whilst in the previous