Algorithm 3.21 Point doubling ($y^2 = x^3 - 3x + b$, Jacobian coordinates)

INPUT: $P = (X_1 : Y_1 : Z_1)$ in Jacobian coordinates on $E/K : y^2 = x^3 - 3x + b$.
OUTPUT: $2P = (X_3 : Y_3 : Z_3)$ in Jacobian coordinates.

1. If $P = \infty$ then return(∞).
2. $T_1 \leftarrow Z_1^2$. $\{T_1 \leftarrow Z_1^2\}$
3. $T_2 \leftarrow X_1 - T_1$. $\{T_2 \leftarrow X_1 - Z_1^2\}$
4. $T_1 \leftarrow X_1 + T_1$. $\{T_1 \leftarrow X_1 + Z_1^2\}$
5. $T_2 \leftarrow T_2 \cdot T_1$. $\{T_2 \leftarrow X_1^2 - Z_1^4\}$
6. $T_2 \leftarrow 3T_2$. $\{T_2 \leftarrow A = 3(X_1 - Z_1^2)(X_1 + Z_1^2)\}$
7. $Y_3 \leftarrow 2Y_1$. $\{Y_3 \leftarrow B = 2Y_1\}$
8. $Z_3 \leftarrow Y_3 \cdot Z_1$. $\{Z_3 \leftarrow BZ_1\}$
9. $Y_3 \leftarrow Y_3^2$. $\{Y_3 \leftarrow C = B^2\}$
10. $T_3 \leftarrow Y_3 \cdot X_1$. $\{T_3 \leftarrow D = CX_1\}$
11. $Y_3 \leftarrow Y_3^2$. $\{Y_3 \leftarrow C^2\}$
12. $Y_3 \leftarrow Y_3/2$. $\{Y_3 \leftarrow C^2/2\}$
13. $X_3 \leftarrow T_2^2$. $\{X_3 \leftarrow A^2\}$
14. $T_1 \leftarrow 2T_3$. $\{T_1 \leftarrow 2D\}$
15. $X_3 \leftarrow X_3 - T_1$. $\{X_3 \leftarrow A^2 - 2D\}$
16. $T_1 \leftarrow T_3 - X_3$. $\{T_1 \leftarrow D - X_3\}$
17. $T_1 \leftarrow T_1 \cdot T_2$. $\{T_1 \leftarrow (D - X_3)A\}$
18. $Y_3 \leftarrow T_1 - Y_3$. $\{Y_3 \leftarrow (D - X_3)A - C^2/2\}$
19. Return($X_3 : Y_3 : Z_3$).

Algorithm 3.22 Point addition ($y^2 = x^3 - 3x + b$, affine-Jacobian coordinates)

INPUT: $P = (X_1 : Y_1 : Z_1)$ in Jacobian coordinates, $Q = (x_2, y_2)$ in affine coordinates on $E/K : y^2 = x^3 - 3x + b$.
OUTPUT: $P + Q = (X_3 : Y_3 : Z_3)$ in Jacobian coordinates.

1. If $Q = \infty$ then return($X_1 : Y_1 : Z_1$).
2. If $P = \infty$ then return($x_2 : y_2 : 1$).
3. $T_1 \leftarrow Z_1^2$. $\{T_1 \leftarrow A = Z_1^2\}$
4. $T_2 \leftarrow T_1 \cdot Z_1$. $\{T_2 \leftarrow B = Z_1 A\}$
5. $T_1 \leftarrow T_1 \cdot x_2$. $\{T_1 \leftarrow C = X_2 A\}$
6. $T_2 \leftarrow T_2 \cdot y_2$. $\{T_2 \leftarrow D = Y_2 B\}$
7. $T_1 \leftarrow T_1 - X_1$. $\{T_1 \leftarrow E = C - X_1\}$
8. $T_2 \leftarrow T_2 - Y_1$. $\{T_2 \leftarrow F = D - Y_1\}$
9. If $T_1 = 0$ then
   - 9.1 If $T_2 = 0$ then use Algorithm 3.21 to compute $(X_3 : Y_3 : Z_3) = 2(x_2 : y_2 : 1)$ and return($X_3 : Y_3 : Z_3$).
   - 9.2 Else return(∞).
10. $Z_3 \leftarrow Z_1 \cdot T_1$. $\{Z_3 \leftarrow Z_1 E\}$

11. $T_3 \leftarrow T_1^2$.	$\{T_3 \leftarrow G = E^2\}$
12. $T_4 \leftarrow T_3 \cdot T_1$.	$\{T_4 \leftarrow H = E^3\}$
13. $T_3 \leftarrow T_3 \cdot X_1$.	$\{T_3 \leftarrow I = X_1 G\}$
14. $T_1 \leftarrow 2T_3$.	$\{T_1 \leftarrow 2I\}$
15. $X_3 \leftarrow T_2^2$.	$\{X_3 \leftarrow F^2\}$
16. $X_3 \leftarrow X_3 - T_1$.	$\{X_3 \leftarrow F^2 - 2I\}$
17. $X_3 \leftarrow X_3 - T_4$.	$\{X_3 \leftarrow F^2 - (H + 2I)\}$
18. $T_3 \leftarrow T_3 - X_3$.	$\{T_3 \leftarrow I - X_3\}$
19. $T_3 \leftarrow T_3 \cdot T_2$.	$\{T_3 \leftarrow F(I - X_3)\}$
20. $T_4 \leftarrow T_4 \cdot Y_1$.	$\{T_4 \leftarrow Y_1 H\}$
21. $Y_3 \leftarrow T_3 - T_4$.	$\{Y_3 \leftarrow F(I - X_3) - Y_1 H\}$
22. Return$(X_3 : Y_3 : Z_3)$.	

The field operation counts for point addition and doubling in various coordinate systems are listed in Table 3.3. The notation $C_1 + C_2 \rightarrow C_3$ means that the points to be added are in C_1 coordinates and C_2 coordinates, while their sum is expressed in C_3 coordinates; for example, $J + A \rightarrow J$ is an addition of points in Jacobian and affine coordinates, with result in Jacobian coordinates. We see that Jacobian coordinates yield the fastest point doubling, while mixed Jacobian-affine coordinates yield the fastest point addition. Also useful in some point multiplication algorithms (see Note 3.43) are mixed Jacobian-Chudnovsky coordinates and mixed Chudnovsky-affine coordinates for point addition.

Doubling		General addition		Mixed coordinates	
$2A \rightarrow A$	$1I, 2M, 2S$	$A + A \rightarrow A$	$1I, 2M, 1S$	$J + A \rightarrow J$	$8M, 3S$
$2P \rightarrow P$	$7M, 3S$	$P + P \rightarrow P$	$12M, 2S$	$J + C \rightarrow J$	$11M, 3S$
$2J \rightarrow J$	$4M, 4S$	$J + J \rightarrow J$	$12M, 4S$	$C + A \rightarrow C$	$8M, 3S$
$2C \rightarrow C$	$5M, 4S$	$C + C \rightarrow C$	$11M, 3S$		

Table 3.3. Operation counts for point addition and doubling on $y^2 = x^3 - 3x + b$. A = affine, P = standard projective, J = Jacobian, C = Chudnovsky, I = inversion, M = multiplication, S = squaring.

Repeated doublings

If consecutive point doublings are to be performed, then Algorithm 3.23 may be slightly faster than repeated use of the doubling formula. By working with $2Y$ until the final step, only one division by 2 is required. A field addition in the loop is eliminated by calculating $3(X - Z^2)(X + Z^2)$ as $3(X^2 - W)$, where $W = Z^4$ is computed at the first doubling and then updated according to $W \leftarrow WY^4$ before each subsequent doubling.

Algorithm 3.27 Left-to-right binary method for point multiplication

INPUT: $k = (k_{t-1}, \ldots, k_1, k_0)_2$, $P \in E(\mathbb{F}_q)$.
OUTPUT: kP.

1. $Q \leftarrow \infty$.
2. For i from $t-1$ downto 0 do
   2.1 $Q \leftarrow 2Q$.
   2.2 If $k_i = 1$ then $Q \leftarrow Q + P$.
3. Return(Q).

The expected number of ones in the binary representation of k is $t/2 \approx m/2$, whence the expected running time of Algorithm 3.27 is approximately $m/2$ point additions and m point doublings, denoted

$$\frac{m}{2}A + mD. \tag{3.15}$$

Let M denote a field multiplication, S a field squaring, and I a field inversion. If affine coordinates (see §3.1.2) are used, then the running time expressed in terms of field operations is

$$2.5mS + 3mM + 1.5mI \tag{3.16}$$

if $\mathbb{F}_q$ has characteristic > 3, and

$$3mM + 1.5mI \tag{3.17}$$

if $\mathbb{F}_q$ is a binary field.

Suppose that $\mathbb{F}_q$ has characteristic > 3. If mixed coordinates (see §3.2.2) are used, then Q is stored in Jacobian coordinates, while P is stored in affine coordinates. Thus the doubling in step 2.1 can be performed using Algorithm 3.21, while the addition in step 2.2 can be performed using Algorithm 3.22. The field operation count of Algorithm 3.27 is then

$$8mM + 5.5mS + (1I + 3M + 1S) \tag{3.18}$$

(one inversion, three multiplications and one squaring are required to convert back to affine coordinates).

Suppose now that $\mathbb{F}_q$ is a binary field. If mixed coordinates (see §3.2.3) are used, then Q is stored in LD projective coordinates, while P can be stored in affine coordinates. Thus the doubling in step 2.1 can be performed using Algorithm 3.24, and the addition in step 2.2 can be performed using Algorithm 3.25. The field operation count of Algorithm 3.27 is then

$$8.5mM + (2M + 1I) \tag{3.19}$$

(one inversion and two multiplications are required to convert back to affine coordinates).