**Fonction Maple 2023 (23.2) permettant de calculer l’addition de deux points d’une courbe elliptique ECC, et la multiplication du résultat de cette somme par un scalaire, en utilisant l’algèbre associée aux courbes elliptiques (EC) :**

ECCAddition := proc(P, Q, a, b, p) local x1, y1, x2, y2, m, xr, yr; x1 := P[1]; y1 := P[2]; x2 := Q[1]; y2 := Q[2]; if P = [0, 0] then return Q; elif Q = [0, 0] then return P; elif P <> Q then m := (y2 - y1)\*mod\_inverse(x2 - x1, p) mod p; else m := (3\*x1^2 + a)\*mod\_inverse(2\*y1, p) mod p; end if; xr := (m^2 - x1 - x2) mod p; yr := (m\*(x1 - xr) - y1) mod p; return [xr, yr]; end proc;

mod\_inverse := proc(a, n) local inv, i; for i to n - 1 do if i\*a mod n = 1 then return i; end if; end do; end proc;

ECCMultiplication := proc(P, scalar, a, b, p) local Q, i; Q := [0, 0]; for i to scalar do Q := ECCAddition(Q, P, a, b, p); end do; print("Multiplication de point ", P, " par le scalaire ", scalar, " :"); print(Q); return Q; end proc;

a := 2;

b := 9;

p := 37;

point\_de\_depart := [15, 11];

scalaire := 5;

ECCMultiplication(point\_de\_depart, scalaire, a, b, p);

**Fonction Maple 2023 (23.2) permettant de calculer la multiplication d’un point d’une courbe elliptique ECC par un scalaire, en utilisant l’algèbre associée aux courbes elliptiques (EC) :**

ECCMultiplication := proc(P, scalar, a, b, p) local Q, i; Q := [0, 0]; for i to scalar do Q := ECCAddition(Q, P, a, b, p); end do; print("Multiplication de point ", P, " par le scalaire ", scalar, " :"); print(Q); return Q; end proc;

a := 2;

b := 9;

p := 37;

point\_de\_depart := [10, 20];

scalaire := 5;

ECCMultiplication(point\_de\_depart, scalaire, a, b, p);

**Fonction Maple 2023 (23.2) permettant de calculer la clé publique AB, qui est un point dans une courbe elliptique ECC, cette clé publique est calculé en utilisant la multiplication d’un autre point de cette courbe elliptique par un scalaire, en utilisant l’algèbre associée aux courbes elliptiques ECC :**

ECCAddition := proc(P, Q, a, b, p) local x1, y1, x2, y2, m, xr, yr; x1 := P[1]; y1 := P[2]; x2 := Q[1]; y2 := Q[2]; if P = [0, 0] then return Q; elif Q = [0, 0] then return P; elif P <> Q then m := (y2 - y1)\*mod\_inverse(x2 - x1, p) mod p; else m := (3\*x1^2 + a)\*mod\_inverse(2\*y1, p) mod p; end if; xr := (m^2 - x1 - x2) mod p; yr := (m\*(x1 - xr) - y1) mod p; return [xr, yr]; end proc;

mod\_inverse := proc(a, n) local inv, i; for i to n - 1 do if i\*a mod n = 1 then return i; end if; end do; end proc;

ECCMultiplication := proc(P, scalar, a, b, p) local Q, i; Q := [0, 0]; for i to scalar do Q := ECCAddition(Q, P, a, b, p); end do; print("Multiplication de point ", P, " par le scalaire ", scalar, " :"); print(Q); return Q; end proc;

a := 2;

b := 9;

p := 37;

point\_de\_depart := [23, 30];

scalaire := 5;

ECCMultiplication(point\_de\_depart, scalaire, a, b, p);

**Fonction Maple 2023 (23.2) permettant de calculer la clé publique BA, qui est un point dans une courbe elliptique ECC, cette clé publique est calculé en utilisant la multiplication d’un autre point de cette courbe elliptique par un scalaire, en utilisant l’algèbre associée aux courbes elliptiques ECC :**

ECCAddition := proc(P, Q, a, b, p) local x1, y1, x2, y2, m, xr, yr; x1 := P[1]; y1 := P[2]; x2 := Q[1]; y2 := Q[2]; if P = [0, 0] then return Q; elif Q = [0, 0] then return P; elif P <> Q then m := (y2 - y1)\*mod\_inverse(x2 - x1, p) mod p; else m := (3\*x1^2 + a)\*mod\_inverse(2\*y1, p) mod p; end if; xr := (m^2 - x1 - x2) mod p; yr := (m\*(x1 - xr) - y1) mod p; return [xr, yr]; end proc;

mod\_inverse := proc(a, n) local inv, i; for i to n - 1 do if i\*a mod n = 1 then return i; end if; end do; end proc;

ECCMultiplication := proc(P, scalar, a, b, p) local Q, i; Q := [0, 0]; for i to scalar do Q := ECCAddition(Q, P, a, b, p); end do; print("Multiplication de point ", P, " par le scalaire ", scalar, " :"); print(Q); return Q; end proc;

a := 2;

b := 9;

p := 37;

point\_de\_depart := [33, 23];

scalaire := 7;

ECCMultiplication(point\_de\_depart, scalaire, a, b, p);