

General Weierstrass Equations

Arthur Freitas Ramos David Barros Hulak
Ruy J. G. B. de Queiroz

October 1, 2026

Abstract

This entry develops the general Weierstrass equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

over commutative rings and fields. It defines the standard invariants, proves $c_4^3 - c_6^2 = 1728\Delta$, formalizes the affine and projective equations and their singular points, and proves over every field that the projective cubic is geometrically nonsingular exactly when $\Delta \neq 0$. Geometric singularities are taken over the algebraic closure, so the statement includes imperfect fields and characteristics 2 and 3. A final bridge specializes the development to the short equation and reuses the AFP *Elliptic Curves Group Law* entry [1]. The definitions and invariant formulas follow the standard treatment in [2]. AI assistance was used for proof engineering. The final definitions, statements, and proofs are checked by Isabelle.

Contents

1	General Weierstrass coefficients	2
2	Standard invariants	3
3	The affine equation	5
4	Homogeneous coordinates	5
5	The projective cubic	7
6	Formal partial derivatives	9
7	Singular points	10
8	The affine chart	11
9	Elimination polynomials	13

10 Elimination identities	14
11 Characteristic two	16
12 Odd characteristic	25
13 The discriminant criterion	35
14 The partial j-invariant	37
15 Specialization to short Weierstrass equations	38
theory <i>Weierstrass-Coefficients</i>	
imports	
<i>HOL-Computational-Algebra.Polynomial</i>	
begin	

1 General Weierstrass coefficients

record *'a weierstrass-coeffs* =

a1 :: *'a*
a2 :: *'a*
a3 :: *'a*
a4 :: *'a*
a6 :: *'a*

lemma *weierstrass-coeffs-eqI*:

fixes *W V* :: *'a weierstrass-coeffs*

assumes *a1 W = a1 V*

and *a2 W = a2 V*

and *a3 W = a3 V*

and *a4 W = a4 V*

and *a6 W = a6 V*

shows *W = V*

using *assms* **by** (*cases W*; *cases V*) *simp*

definition *map-weierstrass-coeffs* ::

(*'a* $\Rightarrow$ *'b*) $\Rightarrow$ *'a weierstrass-coeffs* $\Rightarrow$ *'b weierstrass-coeffs*

where

map-weierstrass-coeffs f W =

(*a1* = *f (a1 W)*, *a2* = *f (a2 W)*, *a3* = *f (a3 W)*,
a4 = *f (a4 W)*, *a6* = *f (a6 W)*)

lemma *map-weierstrass-coeffs-simps* [*simp*]:

a1 (map-weierstrass-coeffs f W) = f (a1 W)

a2 (map-weierstrass-coeffs f W) = f (a2 W)

a3 (map-weierstrass-coeffs f W) = f (a3 W)

a4 (map-weierstrass-coeffs f W) = f (a4 W)

a6 (map-weierstrass-coeffs f W) = f (a6 W)

```

by (simp-all add: map-weierstrass-coeffs-def)

lemma map-weierstrass-coeffs-id [simp]:
  map-weierstrass-coeffs id W = W
by (cases W) (simp add: map-weierstrass-coeffs-def)

lemma map-weierstrass-coeffs-comp:
  map-weierstrass-coeffs f (map-weierstrass-coeffs g W) =
  map-weierstrass-coeffs (f ∘ g) W
by (cases W) (simp add: map-weierstrass-coeffs-def)

definition short-weierstrass-coeffs ::
  'a::zero ⇒ 'a ⇒ 'a weierstrass-coeffs
where
  short-weierstrass-coeffs A B =
  (a1 = 0, a2 = 0, a3 = 0, a4 = A, a6 = B)

lemma short-weierstrass-coeffs-simps [simp]:
  a1 (short-weierstrass-coeffs A B) = 0
  a2 (short-weierstrass-coeffs A B) = 0
  a3 (short-weierstrass-coeffs A B) = 0
  a4 (short-weierstrass-coeffs A B) = A
  a6 (short-weierstrass-coeffs A B) = B
by (simp-all add: short-weierstrass-coeffs-def)

end
theory Weierstrass-Invariants
imports Weierstrass-Coefficients
begin

```

2 Standard invariants

```

definition b2 :: 'a::comm-ring-1 weierstrass-coeffs ⇒ 'a
  where b2 W = a1 W ^ 2 + 4 * a2 W

definition b4 :: 'a::comm-ring-1 weierstrass-coeffs ⇒ 'a
  where b4 W = 2 * a4 W + a1 W * a3 W

definition b6 :: 'a::comm-ring-1 weierstrass-coeffs ⇒ 'a
  where b6 W = a3 W ^ 2 + 4 * a6 W

definition b8 :: 'a::comm-ring-1 weierstrass-coeffs ⇒ 'a
  where
    b8 W =
      a1 W ^ 2 * a6 W + 4 * a2 W * a6 W - a1 W * a3 W * a4 W +
      a2 W * a3 W ^ 2 - a4 W ^ 2

definition c4 :: 'a::comm-ring-1 weierstrass-coeffs ⇒ 'a
  where c4 W = b2 W ^ 2 - 24 * b4 W

```

definition $c6 :: 'a::comm-ring-1 \text{ weierstrass-coeffs} \Rightarrow 'a$
where $c6 \ W = -(b2 \ W^3) + 36 * b2 \ W * b4 \ W - 216 * b6 \ W$

definition $discriminant :: 'a::comm-ring-1 \text{ weierstrass-coeffs} \Rightarrow 'a$
where
 $discriminant \ W =$
 $-(b2 \ W^2) * b8 \ W - 8 * b4 \ W^3 - 27 * b6 \ W^2 +$
 $9 * b2 \ W * b4 \ W * b6 \ W$

lemma $b2-b6-minus-b4-squared$:
 $b2 \ W * b6 \ W - b4 \ W^2 = 4 * b8 \ W$

proof –
obtain $A1 \ A2 \ A3 \ A4 \ A6$ **where** W :
 $W = (a1 = A1, a2 = A2, a3 = A3, a4 = A4, a6 = A6)$
by $(cases \ W) \ blast$
show $?thesis$
unfolding $W \ b2-def \ b4-def \ b6-def \ b8-def$
by $(simp \ add: \ algebra-simps \ power2-eq-square)$
qed

theorem $c4-c6-discriminant-identity$:
 $c4 \ W^3 - c6 \ W^2 = 1728 * discriminant \ W$

proof –
obtain $A1 \ A2 \ A3 \ A4 \ A6$ **where** W :
 $W = (a1 = A1, a2 = A2, a3 = A3, a4 = A4, a6 = A6)$
by $(cases \ W) \ blast$
show $?thesis$
unfolding $W \ b2-def \ b4-def \ b6-def \ b8-def \ c4-def \ c6-def \ discriminant-def$
by $(simp \ add: \ algebra-simps \ power2-eq-square \ power3-eq-cube)$
qed

lemma $four-discriminant-expanded$:
 $4 * discriminant \ W =$
 $-(b2 \ W^3) * b6 \ W + b2 \ W^2 * b4 \ W^2 -$
 $32 * b4 \ W^3 - 108 * b6 \ W^2 +$
 $36 * b2 \ W * b4 \ W * b6 \ W$

proof –
obtain $A1 \ A2 \ A3 \ A4 \ A6$ **where** W :
 $W = (a1 = A1, a2 = A2, a3 = A3, a4 = A4, a6 = A6)$
by $(cases \ W) \ blast$
show $?thesis$
unfolding $W \ b2-def \ b4-def \ b6-def \ b8-def \ discriminant-def$
by $(simp \ add: \ algebra-simps \ power2-eq-square \ power3-eq-cube)$
qed

end
theory $Weierstrass-Affine-Curve$
imports $Weierstrass-Invariants$

begin

3 The affine equation

definition *weierstrass-affine-poly* ::

'a::comm-ring-1 *weierstrass-coeffs* $\Rightarrow$ *'a* $\Rightarrow$ *'a* $\Rightarrow$ *'a*

where

weierstrass-affine-poly *W x y* =

$y^2 + a1 \cdot W \cdot x \cdot y + a3 \cdot W \cdot y -$
 $(x^3 + a2 \cdot W \cdot x^2 + a4 \cdot W \cdot x + a6 \cdot W)$

definition *on-weierstrass-affine* ::

'a::comm-ring-1 *weierstrass-coeffs* $\Rightarrow$ *'a* $\Rightarrow$ *'a* $\Rightarrow$ *bool*

where

on-weierstrass-affine *W x y* $\longleftrightarrow$

$y^2 + a1 \cdot W \cdot x \cdot y + a3 \cdot W \cdot y =$
 $x^3 + a2 \cdot W \cdot x^2 + a4 \cdot W \cdot x + a6 \cdot W$

lemma *on-weierstrass-affine-iff-poly-zero*:

on-weierstrass-affine *W x y* $\longleftrightarrow$

weierstrass-affine-poly *W x y* = 0

unfolding *on-weierstrass-affine-def weierstrass-affine-poly-def*

by *simp*

end

theory *Weierstrass-Projective-Curve*

imports *Weierstrass-Affine-Curve*

begin

4 Homogeneous coordinates

type-synonym *'a weierstrass-projective-coords* = *'a* $\times$ *'a* $\times$ *'a*

definition *scale-projective* ::

'a::times $\Rightarrow$ *'a weierstrass-projective-coords* $\Rightarrow$

'a weierstrass-projective-coords

where

scale-projective *u* = ($\lambda(X, Y, Z). (u * X, u * Y, u * Z)$)

lemma *scale-projective-simps* [*simp*]:

scale-projective *u* (*X*, *Y*, *Z*) = (*u* * *X*, *u* * *Y*, *u* * *Z*)

by (*simp add: scale-projective-def*)

definition *projective-coords-valid* ::

'a::zero *weierstrass-projective-coords* $\Rightarrow$ *bool*

where

projective-coords-valid = ($\lambda(X, Y, Z). X \neq 0 \vee Y \neq 0 \vee Z \neq 0$)

definition *projectively-equivalent* ::
 ['a::field weierstrass-projective-coords](#) $\Rightarrow$
 ['a weierstrass-projective-coords](#) $\Rightarrow$ *bool*
where
projectively-equivalent *p q* $\longleftrightarrow$
 ($\exists u. u \neq 0 \wedge q = \text{scale-projective } u \ p$)

lemma *scale-projective-one* [*simp*]:
scale-projective 1 *p* = *p*
for *p* :: ['a::monoid-mult weierstrass-projective-coords](#)
by (*induct* *p* *rule*: *prod-induct3*) *simp*

lemma *scale-projective-mult*:
scale-projective *u* (*scale-projective* *v* *p*) = *scale-projective* (*u* * *v*) *p*
for *p* :: ['a::monoid-mult weierstrass-projective-coords](#)
by (*induct* *p* *rule*: *prod-induct3*) (*simp* *add*: *mult.assoc*)

lemma *projective-coords-valid-scale-iff*:
projective-coords-valid (*scale-projective* *u* *p*) $\longleftrightarrow$
 $u \neq 0 \wedge \text{projective-coords-valid } p$
for *u* :: ['a::field](#)
by (*induct* *p* *rule*: *prod-induct3*)
 (*auto simp*: *projective-coords-valid-def*)

lemma *projectively-equivalent-refl* [*simp*]:
projectively-equivalent *p* *p*
unfolding *projectively-equivalent-def*
by (*rule* *exI*[*of* - 1]) *simp*

lemma *projectively-equivalent-sym*:
projectively-equivalent *p* *q* $\implies$ *projectively-equivalent* *q* *p*
unfolding *projectively-equivalent-def*
proof –
assume $\exists u. u \neq 0 \wedge q = \text{scale-projective } u \ p$
then obtain *u* **where** *u*: $u \neq 0 \wedge q = \text{scale-projective } u \ p$
 by *blast*
show $\exists v. v \neq 0 \wedge p = \text{scale-projective } v \ q$
proof (*intro* *exI*[*of* - *inverse* *u*] *conjI*)
 show *inverse* *u* $\neq 0$
 using *u* **by** *simp*
 show *p* = *scale-projective* (*inverse* *u*) *q*
 unfolding *u*(2) *scale-projective-mult*
 using *u* **by** *simp*
qed
qed

lemma *projectively-equivalent-trans*:
projectively-equivalent *p* *q* $\implies$
projectively-equivalent *q* *r* $\implies$

$\text{projectively-equivalent } p \ r$
unfolding $\text{projectively-equivalent-def}$
proof –
assume $\exists u. u \neq 0 \wedge q = \text{scale-projective } u \ p$
and $\exists v. v \neq 0 \wedge r = \text{scale-projective } v \ q$
then obtain $u \ v$ **where** uv :
 $u \neq 0 \ v \neq 0$
 $q = \text{scale-projective } u \ p \ r = \text{scale-projective } v \ q$
by *blast*
show $\exists w. w \neq 0 \wedge r = \text{scale-projective } w \ p$
using uv
by (*intro exI[of - v * u]*) (*simp add: scale-projective-mult*)
qed

5 The projective cubic

definition $\text{weierstrass-projective-poly} ::$
 $'a :: \text{comm-ring-1 weierstrass-coeffs} \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where
 $\text{weierstrass-projective-poly } W \ X \ Y \ Z =$
 $Y^2 * Z + a1 \ W * X * Y * Z + a3 \ W * Y * Z^2 -$
 $(X^3 + a2 \ W * X^2 * Z + a4 \ W * X * Z^2 + a6 \ W * Z^3)$

definition $\text{on-projective-weierstrass} ::$
 $'a :: \text{comm-ring-1 weierstrass-coeffs} \Rightarrow$
 $'a \ \text{weierstrass-projective-coords} \Rightarrow \text{bool}$
where
 $\text{on-projective-weierstrass } W =$
 $(\lambda(X, Y, Z). \text{projective-coords-valid } (X, Y, Z) \wedge$
 $\text{weierstrass-projective-poly } W \ X \ Y \ Z = 0)$

lemma $\text{weierstrass-projective-poly-scale}:$
 $\text{weierstrass-projective-poly } W \ (u * X) \ (u * Y) \ (u * Z) =$
 $u^3 * \text{weierstrass-projective-poly } W \ X \ Y \ Z$
by (*simp add: weierstrass-projective-poly-def algebra-simps*
 $\text{power2-eq-square power3-eq-cube}$)

lemma $\text{on-projective-weierstrass-scale-iff}:$
fixes $u :: 'a :: \text{field}$
assumes $u \neq 0$
shows $\text{on-projective-weierstrass } W \ (\text{scale-projective } u \ p) \longleftrightarrow$
 $\text{on-projective-weierstrass } W \ p$
using *assms*
by (*induct p rule: prod-induct3*)
 $(\text{simp add: on-projective-weierstrass-def projective-coords-valid-def}$
 $\text{weierstrass-projective-poly-scale assms})$

definition $\text{affine-to-projective} ::$

$'a::one \Rightarrow 'a \Rightarrow 'a$ *weierstrass-projective-coords*
where
affine-to-projective $x\ y = (x, y, 1)$

lemma *weierstrass-projective-poly-affine* [simp]:
weierstrass-projective-poly $W\ x\ y\ 1 = \text{weierstrass-affine-poly}\ W\ x\ y$
by (simp add: *weierstrass-projective-poly-def weierstrass-affine-poly-def*)

theorem *on-projective-weierstrass-affine*:
on-projective-weierstrass $W\ (\text{affine-to-projective}\ x\ y) \longleftrightarrow$
on-weierstrass-affine $W\ x\ y$
by (simp add: *affine-to-projective-def on-projective-weierstrass-def*
projective-coords-valid-def on-weierstrass-affine-iff-poly-zero)

theorem *on-projective-weierstrass-chart*:
fixes $X\ Y\ Z :: 'a::field$
assumes $Z \neq 0$
shows *on-projective-weierstrass* $W\ (X, Y, Z) \longleftrightarrow$
on-weierstrass-affine $W\ (X / Z)\ (Y / Z)$
proof –
have *invZ*: *inverse* $Z \neq 0$
using *assms* **by** *simp*
have *scale*:
scale-projective (*inverse* Z) $(X, Y, Z) =$
affine-to-projective $(X / Z)\ (Y / Z)$
using *assms*
by (simp add: *affine-to-projective-def field-class.field-divide-inverse*)
have *scaled*:
on-projective-weierstrass W
 $(\text{affine-to-projective}\ (X / Z)\ (Y / Z)) \longleftrightarrow$
on-projective-weierstrass $W\ (X, Y, Z)$
using *on-projective-weierstrass-scale-iff*[*OF invZ, of W (X, Y, Z)*]
unfolding *scale* .
show ?thesis
using *scaled on-projective-weierstrass-affine*[*of W X / Z Y / Z*]
by *blast*
qed

definition *weierstrass-infinity* ::
 $'a::zero\ neq\ one$ *weierstrass-projective-coords*
where
weierstrass-infinity = $(0, 1, 0)$

lemma *weierstrass-infinity-on-curve* [simp]:
on-projective-weierstrass $W\ \text{weierstrass-infinity}$
by (simp add: *on-projective-weierstrass-def projective-coords-valid-def*
weierstrass-infinity-def weierstrass-projective-poly-def)

lemma *on-projective-weierstrass-at-infinity-iff*:

```

on-projective-weierstrass W (X, Y, 0)  $\longleftrightarrow$ 
  X = 0  $\wedge$  Y  $\neq$  0
for X Y :: 'a::field
by (auto simp: on-projective-weierstrass-def projective-coords-valid-def
    weierstrass-projective-poly-def)

theorem projective-weierstrass-at-infinity:
  assumes on-projective-weierstrass W (X, Y, 0)
  shows projectively-equivalent (X, Y, 0) weierstrass-infinity
proof -
  from assms have X: X = 0 and Y: Y  $\neq$  0
  by (simp-all add: on-projective-weierstrass-at-infinity-iff)
  show ?thesis
  unfolding projectively-equivalent-def
proof (intro exI[of - inverse Y] conjI)
  show inverse Y  $\neq$  0
  using Y by simp
  show weierstrass-infinity =
    scale-projective (inverse Y) (X, Y, 0)
  using X Y by (simp add: weierstrass-infinity-def)
qed
qed

end
theory Weierstrass-Singularities
imports Weierstrass-Projective-Curve
begin

```

6 Formal partial derivatives

```

definition weierstrass-projective-dx ::
  'a::comm-ring-1 weierstrass-coeffs  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a
where
  weierstrass-projective-dx W X Y Z =
    a1 W * Y * Z - 3 * X ^ 2 - 2 * a2 W * X * Z - a4 W * Z ^ 2

definition weierstrass-projective-dy ::
  'a::comm-ring-1 weierstrass-coeffs  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a
where
  weierstrass-projective-dy W X Y Z =
    2 * Y * Z + a1 W * X * Z + a3 W * Z ^ 2

definition weierstrass-projective-dz ::
  'a::comm-ring-1 weierstrass-coeffs  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a
where
  weierstrass-projective-dz W X Y Z =
    Y ^ 2 + a1 W * X * Y + 2 * a3 W * Y * Z -
    a2 W * X ^ 2 - 2 * a4 W * X * Z - 3 * a6 W * Z ^ 2

```

lemma *weierstrass-projective-dx-scale*:

$$\text{weierstrass-projective-dx } W (u * X) (u * Y) (u * Z) =$$

$$u^2 * \text{weierstrass-projective-dx } W X Y Z$$
by (*simp add: weierstrass-projective-dx-def algebra-simps power2-eq-square*)

lemma *weierstrass-projective-dy-scale*:

$$\text{weierstrass-projective-dy } W (u * X) (u * Y) (u * Z) =$$

$$u^2 * \text{weierstrass-projective-dy } W X Y Z$$
by (*simp add: weierstrass-projective-dy-def algebra-simps power2-eq-square*)

lemma *weierstrass-projective-dz-scale*:

$$\text{weierstrass-projective-dz } W (u * X) (u * Y) (u * Z) =$$

$$u^2 * \text{weierstrass-projective-dz } W X Y Z$$
by (*simp add: weierstrass-projective-dz-def algebra-simps power2-eq-square*)

lemma *weierstrass-projective-euler*:

$$X * \text{weierstrass-projective-dx } W X Y Z +$$

$$Y * \text{weierstrass-projective-dy } W X Y Z +$$

$$Z * \text{weierstrass-projective-dz } W X Y Z =$$

$$3 * \text{weierstrass-projective-poly } W X Y Z$$
by (*simp add: weierstrass-projective-dx-def weierstrass-projective-dy-def*
weierstrass-projective-dz-def weierstrass-projective-poly-def
algebra-simps power2-eq-square power3-eq-cube)

7 Singular points

definition *singular-projective-point* ::

$$'a::\text{field} \text{ weierstrass-coeffs} \Rightarrow$$

$$'a \text{ weierstrass-projective-coords} \Rightarrow \text{bool}$$

where

singular-projective-point $W =$

$$(\lambda(X, Y, Z).$$

$$\text{projective-coords-valid } (X, Y, Z) \wedge$$

$$\text{weierstrass-projective-poly } W X Y Z = 0 \wedge$$

$$\text{weierstrass-projective-dx } W X Y Z = 0 \wedge$$

$$\text{weierstrass-projective-dy } W X Y Z = 0 \wedge$$

$$\text{weierstrass-projective-dz } W X Y Z = 0)$$

lemma *singular-projective-point-on-curve*:

$$\text{singular-projective-point } W p \implies \text{on-projective-weierstrass } W p$$
by (*induct p rule: prod-induct3*)
(simp add: singular-projective-point-def on-projective-weierstrass-def)

theorem *singular-projective-point-scale-iff*:
fixes $u :: 'a::\text{field}$
assumes $u \neq 0$
shows $\text{singular-projective-point } W (\text{scale-projective } u p) \longleftrightarrow$
 $\text{singular-projective-point } W p$
using *assms*

by (*induct p rule: prod-induct3*)
 (*simp add: singular-projective-point-def projective-coords-valid-def*
weierstrass-projective-poly-scale weierstrass-projective-dx-scale
weierstrass-projective-dy-scale weierstrass-projective-dz-scale assms)

theorem *point-at-infinity-nonsingular*:
 $\neg$ *singular-projective-point* *W* *weierstrass-infinity*
by (*simp add: singular-projective-point-def weierstrass-infinity-def*
projective-coords-valid-def weierstrass-projective-dz-def)

lemma *singular-projective-point-imp-z-nonzero*:
 assumes *singular-projective-point* *W* (*X*, *Y*, *Z*)
 shows $Z \neq 0$
proof
 assume *Z0*: $Z = 0$
 with *assms* have $X = 0$ and $Y = 0$
 by (*auto simp: singular-projective-point-def projective-coords-valid-def*
weierstrass-projective-poly-def weierstrass-projective-dz-def)
 with *assms Z0* show *False*
 by (*simp add: singular-projective-point-def projective-coords-valid-def*)
qed

8 The affine chart

definition *weierstrass-affine-dx* ::
 $'a::\text{comm-ring-1}$ *weierstrass-coeffs* $\Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where
weierstrass-affine-dx *W x y* =
 $a1 \ W * y - 3 * x^2 - 2 * a2 \ W * x - a4 \ W$

definition *weierstrass-affine-dy* ::
 $'a::\text{comm-ring-1}$ *weierstrass-coeffs* $\Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where
weierstrass-affine-dy *W x y* = $2 * y + a1 \ W * x + a3 \ W$

definition *singular-affine-point* ::
 $'a::\text{field}$ *weierstrass-coeffs* $\Rightarrow 'a \Rightarrow 'a \Rightarrow \text{bool}$
where
singular-affine-point *W x y* $\longleftrightarrow$
weierstrass-affine-poly *W x y* = 0 $\wedge$
weierstrass-affine-dx *W x y* = 0 $\wedge$
weierstrass-affine-dy *W x y* = 0

lemma *weierstrass-projective-dx-affine* [*simp*]:
weierstrass-projective-dx *W x y* 1 = *weierstrass-affine-dx* *W x y*
by (*simp add: weierstrass-projective-dx-def weierstrass-affine-dx-def*)

lemma *weierstrass-projective-dy-affine* [*simp*]:
weierstrass-projective-dy *W x y* 1 = *weierstrass-affine-dy* *W x y*

```

by (simp add: weierstrass-projective-dy-def weierstrass-affine-dy-def)

lemma weierstrass-projective-dz-affine:
  weierstrass-projective-dz W x y 1 =
    3 * weierstrass-affine-poly W x y -
    x * weierstrass-affine-dx W x y -
    y * weierstrass-affine-dy W x y
by (simp add: weierstrass-projective-dz-def weierstrass-affine-poly-def
  weierstrass-affine-dx-def weierstrass-affine-dy-def
  algebra-simps power2-eq-square power3-eq-cube)

lemma singular-projective-affine-iff:
  singular-projective-point W (affine-to-projective x y)  $\longleftrightarrow$ 
  singular-affine-point W x y
by (auto simp: singular-projective-point-def singular-affine-point-def
  affine-to-projective-def projective-coords-valid-def
  weierstrass-projective-dz-affine)

theorem projective-singular-iff-affine-singular:
  ( $\exists p.$  singular-projective-point W p)  $\longleftrightarrow$ 
  ( $\exists x y.$  singular-affine-point W x y)
proof
  assume  $\exists p.$  singular-projective-point W p
  then obtain X Y Z where singular: singular-projective-point W (X, Y, Z)
  by (metis prod-cases3)
  have Z: Z  $\neq$  0
  using singular by (rule singular-projective-point-imp-z-nonzero)
  have invZ: inverse Z  $\neq$  0
  using Z by simp
  have scale:
    scale-projective (inverse Z) (X, Y, Z) =
    affine-to-projective (X / Z) (Y / Z)
  using Z
  by (simp add: affine-to-projective-def field-class.field-divide-inverse)
  have singular-projective-point W
    (affine-to-projective (X / Z) (Y / Z))
  using singular-projective-point-scale-iff[OF invZ, of W (X, Y, Z)]
  singular
  unfolding scale
  by blast
  then show  $\exists x y.$  singular-affine-point W x y
  using singular-projective-affine-iff by blast
next
  assume  $\exists x y.$  singular-affine-point W x y
  then obtain x y where singular-affine-point W x y
  by blast
  then show  $\exists p.$  singular-projective-point W p
  using singular-projective-affine-iff by blast
qed

```

definition *projective-weierstrass-nonsingular-over* ::

'a::field weierstrass-coeffs $\Rightarrow$ *bool*

where

projective-weierstrass-nonsingular-over *W* $\longleftrightarrow$

$\neg (\exists p. \text{singular-projective-point } W p)$

lemma *projective-weierstrass-nonsingular-over-iff*:

projective-weierstrass-nonsingular-over *W* $\longleftrightarrow$

$\neg (\exists x y. \text{singular-affine-point } W x y)$

unfolding *projective-weierstrass-nonsingular-over-def*

using *projective-singular-iff-affine-singular* **by** *blast*

9 Elimination polynomials

definition *weierstrass-g* ::

'a::comm-ring-1 weierstrass-coeffs $\Rightarrow$ *'a* $\Rightarrow$ *'a*

where

weierstrass-g *W* *x* =

$4 * x^3 + b2 \ W * x^2 + 2 * b4 \ W * x + b6 \ W$

definition *weierstrass-h* ::

'a::comm-ring-1 weierstrass-coeffs $\Rightarrow$ *'a* $\Rightarrow$ *'a*

where

weierstrass-h *W* *x* = $6 * x^2 + b2 \ W * x + b4 \ W$

lemma *affine-dy-squared-minus-g*:

weierstrass-affine-dy *W* *x* *y* $^2 - \text{weierstrass-g } W x =$

$4 * \text{weierstrass-affine-poly } W x y$

proof –

obtain *A1 A2 A3 A4 A6* **where** *W*:

$W = (a1 = A1, a2 = A2, a3 = A3, a4 = A4, a6 = A6)$

by (*cases* *W*) *blast*

show *?thesis*

unfolding *W* *weierstrass-affine-dy-def* *weierstrass-g-def*

weierstrass-affine-poly-def *b2-def* *b4-def* *b6-def*

by (*simp* *add: algebra-simps power2-eq-square power3-eq-cube*)

qed

lemma *two-affine-dx-minus-a1-dy*:

$2 * \text{weierstrass-affine-dx } W x y -$

$a1 \ W * \text{weierstrass-affine-dy } W x y =$

$-\text{weierstrass-h } W x$

proof –

obtain *A1 A2 A3 A4 A6* **where** *W*:

$W = (a1 = A1, a2 = A2, a3 = A3, a4 = A4, a6 = A6)$

by (*cases* *W*) *blast*

show *?thesis*

unfolding *W* *weierstrass-affine-dx-def* *weierstrass-affine-dy-def*

```

      weierstrass-h-def b2-def b4-def
    by (simp add: algebra-simps power2-eq-square)
qed

end
theory Weierstrass-Discriminant-Criterion
  imports
    Weierstrass-Singularities
    HOL-Algebra.Algebraic-Closure-Type
begin

```

10 Elimination identities

lemma *four-discriminant-bezout*:

```

  4 * discriminant W =
    - (b2 W ^ 3 + 6 * b2 W ^ 2 * x - 30 * b2 W * b4 W -
      144 * b4 W * x + 108 * b6 W) * weierstrass-g W x +
    (b2 W ^ 3 * x + b2 W ^ 2 * b4 W + 4 * b2 W ^ 2 * x ^ 2 -
      28 * b2 W * b4 W * x + 6 * b2 W * b6 W -
      32 * b4 W ^ 2 - 96 * b4 W * x ^ 2 + 72 * b6 W * x) *
    weierstrass-h W x
proof -
  note expanded = four-discriminant-expanded[of W]
  show ?thesis
    unfolding weierstrass-g-def weierstrass-h-def
    using expanded
    by (simp add: algebra-simps power2-eq-square power3-eq-cube)
qed

```

lemma *singular-affine-imp-g-zero*:

```

  assumes singular-affine-point W x y
  shows weierstrass-g W x = 0
proof -
  from assms have F: weierstrass-affine-poly W x y = 0
    and dy: weierstrass-affine-dy W x y = 0
    by (simp-all add: singular-affine-point-def)
  note identity = affine-dy-squared-minus-g[of W x y]
  with F dy show ?thesis
    by simp
qed

```

lemma *singular-affine-imp-h-zero*:

```

  assumes singular-affine-point W x y
  shows weierstrass-h W x = 0
proof -
  from assms have dx: weierstrass-affine-dx W x y = 0
    and dy: weierstrass-affine-dy W x y = 0
    by (simp-all add: singular-affine-point-def)
  note identity = two-affine-dx-minus-a1-dy[of W x y]

```

```

with dx dy show ?thesis
  by simp
qed

```

```

lemma twice-negative-half:
  fixes z :: 'a::field
  assumes (2 :: 'a) ≠ 0
  shows 2 * (- z / 2) + z = 0
  using assms by simp

```

```

lemma four-ne-zero-of-two-ne-zero:
  fixes dummy :: 'a::field
  assumes two: (2 :: 'a) ≠ 0
  shows (4 :: 'a) ≠ 0
proof
  assume four: (4 :: 'a) = 0
  have product: (2 :: 'a) * 2 = 0
  proof -
    have (2 :: 'a) * 2 = 4
    by simp
    also have ... = 0
    by (rule four)
    finally show ?thesis .
  qed
  have (2 :: 'a) = 0 ∨ (2 :: 'a) = 0
    using product by (simp only: mult-eq-0-iff)
  with two show False
  by blast
qed

```

```

lemma affine-singular-of-g-h:
  fixes W :: 'a::field weierstrass-coeffs
  assumes two: (2 :: 'a) ≠ 0
    and g: weierstrass-g W x = 0
    and h: weierstrass-h W x = 0
  shows singular-affine-point W x (- (a1 W * x + a3 W) / 2)
proof -
  let ?y = - (a1 W * x + a3 W) / 2
  have dy: weierstrass-affine-dy W x ?y = 0
    using twice-negative-half[OF two, of a1 W * x + a3 W]
    by (simp only: weierstrass-affine-dy-def add.assoc)
  have four: (4 :: 'a) ≠ 0
    by (rule four-ne-zero-of-two-ne-zero[OF two])
  note F-identity = affine-dy-squared-minus-g[of W x ?y]
  have F: weierstrass-affine-poly W x ?y = 0
    using F-identity dy g four by simp
  note dx-identity = two-affine-dx-minus-a1-dy[of W x ?y]
  have dx: weierstrass-affine-dx W x ?y = 0
    using dx-identity dy h two by simp

```

```

show ?thesis
  unfolding singular-affine-point-def
  using F dx dy by blast
qed

lemma singular-affine-imp-discriminant-zero-odd:
  fixes W :: 'a::field weierstrass-coeffs
  assumes two: (2 :: 'a) ≠ 0
  and singular: singular-affine-point W x y
  shows discriminant W = 0
proof -
  have g: weierstrass-g W x = 0
    using singular by (rule singular-affine-imp-g-zero)
  have h: weierstrass-h W x = 0
    using singular by (rule singular-affine-imp-h-zero)
  have four: (4 :: 'a) ≠ 0
    by (rule four-ne-zero-of-two-ne-zero[OF two])
  have 4 * discriminant W = 0
    using four-discriminant-bezout[of W x] g h by simp
  with four show ?thesis
    by simp
qed

```

11 Characteristic two

```

lemma char-two-three:
  fixes x :: 'a::comm-ring-1
  assumes two: (2 :: 'a) = 0
  shows (3 :: 'a) = 1
proof -
  have (3 :: 'a) = 2 + 1
    by simp
  also have ... = 1
    by (simp add: two)
  finally show ?thesis .
qed

```

```

lemma char-two-four:
  fixes x :: 'a::comm-ring-1
  assumes two: (2 :: 'a) = 0
  shows (4 :: 'a) = 0
proof -
  have (4 :: 'a) = 2 + 2
    by simp
  also have ... = 0
    by (simp add: two)
  finally show ?thesis .
qed

```

```

lemma char-two-eight:
  fixes x :: 'a::comm-ring-1
  assumes two: (2 :: 'a) = 0
  shows (8 :: 'a) = 0
proof -
  have (8 :: 'a) = 4 + 4
    by simp
  also have ... = 0
    by (simp add: char-two-four[OF two])
  finally show ?thesis .
qed

```

```

lemma char-two-nine:
  fixes x :: 'a::comm-ring-1
  assumes two: (2 :: 'a) = 0
  shows (9 :: 'a) = 1
proof -
  have (9 :: 'a) = 8 + 1
    by simp
  also have ... = 1
    by (simp add: char-two-eight[OF two])
  finally show ?thesis .
qed

```

```

lemma char-two-twenty-seven:
  fixes x :: 'a::comm-ring-1
  assumes two: (2 :: 'a) = 0
  shows (27 :: 'a) = 1
proof -
  have (27 :: 'a) = 2 * 13 + 1
    by simp
  also have ... = 1
    by (simp add: two)
  finally show ?thesis .
qed

```

```

lemma char-two-neg:
  fixes x :: 'a::comm-ring-1
  assumes two: (2 :: 'a) = 0
  shows - x = x
proof -
  have x + x = 2 * x
    by (simp add: algebra-simps)
  also have ... = 0
    by (simp add: two)
  finally show ?thesis
    by (simp only: neg-eq-iff-add-eq-0)
qed

```

```

lemma b2-char-two:
  fixes  $W :: 'a::comm-ring-1\ weierstrass-coeffs$ 
  assumes  $two: (2 :: 'a) = 0$ 
  shows  $b2\ W = a1\ W^2$ 
  by (simp add: b2-def char-two-four[OF two])

lemma b4-char-two:
  fixes  $W :: 'a::comm-ring-1\ weierstrass-coeffs$ 
  assumes  $two: (2 :: 'a) = 0$ 
  shows  $b4\ W = a1\ W * a3\ W$ 
  by (simp add: b4-def two)

lemma b6-char-two:
  fixes  $W :: 'a::comm-ring-1\ weierstrass-coeffs$ 
  assumes  $two: (2 :: 'a) = 0$ 
  shows  $b6\ W = a3\ W^2$ 
  by (simp add: b6-def char-two-four[OF two])

lemma b8-char-two:
  fixes  $W :: 'a::comm-ring-1\ weierstrass-coeffs$ 
  assumes  $two: (2 :: 'a) = 0$ 
  shows  $b8\ W =$ 
     $a1\ W^2 * a6\ W + a1\ W * a3\ W * a4\ W +$ 
     $a2\ W * a3\ W^2 + a4\ W^2$ 
  by (simp add: b8-def char-two-four[OF two] char-two-neg[OF two])

lemma discriminant-char-two-expanded:
  fixes  $W :: 'a::comm-ring-1\ weierstrass-coeffs$ 
  assumes  $two: (2 :: 'a) = 0$ 
  shows discriminant  $W =$ 
     $a1\ W^6 * a6\ W + a1\ W^5 * a3\ W * a4\ W +$ 
     $a1\ W^4 * a2\ W * a3\ W^2 + a1\ W^4 * a4\ W^2 +$ 
     $a3\ W^4 + a1\ W^3 * a3\ W^3$ 
proof -
  obtain  $A1\ A2\ A3\ A4\ A6$  where  $W:$ 
     $W = (a1 = A1, a2 = A2, a3 = A3, a4 = A4, a6 = A6)$ 
  by (cases W) blast
  have power-five:  $\bigwedge x :: 'a. x^5 = x * x^4$ 
  proof -
    fix  $x :: 'a$ 
    have  $x^{Suc\ 4} = x * x^4$ 
      by (rule power-Suc)
    then show  $x^5 = x * x^4$ 
      by simp
  qed
  have power-four:  $\bigwedge x :: 'a. x^4 = x * x * x * x$ 
  proof -
    fix  $x :: 'a$ 
    have  $x^{Suc\ 3} = x * x^3$ 

```

```

    by (rule power-Suc)
  then show  $x^4 = x * x * x * x$ 
    by (simp add: power3-eq-cube)
qed
have power-six:  $\bigwedge x :: 'a. x^6 = x * x * x * x * x * x$ 
proof -
  fix x :: 'a
  have  $x^{Suc\ 5} = x * x^5$ 
    by (rule power-Suc)
  then show  $x^6 = x * x * x * x * x * x$ 
    by (simp add: power-five power-four)
qed
have reduced:
  discriminant W =
    ( $a1\ W^2$ )2 * b8 W + ( $a3\ W^2$ )2 +
     $a1\ W^2 * (a1\ W * a3\ W) * a3\ W^2$ 
  unfolding discriminant-def
  apply (simp add: b2-char-two[OF two] b4-char-two[OF two]
    b6-char-two[OF two] char-two-eight[OF two]
    char-two-nine[OF two] char-two-twenty-seven[OF two]
    char-two-neg[OF two] algebra-simps)
  using two
  apply simp
  done
show ?thesis
proof (rule trans)
  show discriminant W =
    ( $a1\ W^2$ )2 * b8 W + ( $a3\ W^2$ )2 +
     $a1\ W^2 * (a1\ W * a3\ W) * a3\ W^2$ 
    by (rule reduced)
  show ( $a1\ W^2$ )2 * b8 W + ( $a3\ W^2$ )2 +
     $a1\ W^2 * (a1\ W * a3\ W) * a3\ W^2 =$ 
     $a1\ W^6 * a6\ W + a1\ W^5 * a3\ W * a4\ W +$ 
     $a1\ W^4 * a2\ W * a3\ W^2 + a1\ W^4 * a4\ W^2 +$ 
     $a3\ W^4 + a1\ W^3 * a3\ W^3$ 
    apply (subst b8-char-two[OF two, of W])
  unfolding W
  by (simp add: power-four power-five power-six
    power2-eq-square power3-eq-cube algebra-simps)
qed
qed

lemma discriminant-char-two-a1-zero:
  fixes W :: 'a::field weierstrass-coeffs
  assumes two: (2 :: 'a) = 0
  and a1:  $a1\ W = 0$ 
  shows discriminant W =  $a3\ W^4$ 
proof -
  show ?thesis

```

using *discriminant-char-two-expanded*[*OF two, of W*] *a1*
 by *simp*
 qed

lemma *scaled-power-eq*:
 fixes $q\ x\ n :: 'a::comm-semiring-1$
 assumes $q * x = n$
 shows $q^k * x^k = n^k$
proof –
 have $(q * x)^k = n^k$
 using *assms* by *simp*
 then show *?thesis*
 by (*simp only: power-mult-distrib*)
 qed

lemma *scale-linear-six*:
 fixes $a\ c\ x :: 'a::comm-ring-1$
 shows $a^6 * (c * x) = a^5 * c * (a * x)$
 by (*simp add: power-numeral-reduce algebra-simps*)

lemma *diff-rearrange*:
 fixes $a\ x\ y\ c :: 'a::comm-ring-1$
 assumes $a * y - x^2 - c = 0$
 shows $y * a = x^2 + c$
proof –
 have $a * y = x^2 + c$
proof –
 from *assms* have $a * y - (x^2 + c) = 0$
 by (*subst diff-diff-add [symmetric]*)
 then show *?thesis*
 by (*rule iffD2 [OF eq-iff-diff-eq-0]*)
 qed
 then show *?thesis*
 by (*subst mult.commute*)
 qed

lemma *char-two-affine-certificate-raw*:
 fixes $A1\ A2\ A3\ A4\ A6 :: 'a::field$
 assumes *two*: $(2 :: 'a) = 0$
 and *A1*: $A1 \neq 0$
 shows $A1^6 * (((((A3 / A1)^2 + A4) / A1)^2 + A1 * (A3 / A1) * (((A3 / A1)^2 + A4) / A1) + A3 * (((A3 / A1)^2 + A4) / A1)) - ((A3 / A1)^3 + A2 * (A3 / A1)^2 + A4 * (A3 / A1) + A6)) = A1^6 * A6 + A1^5 * A3 * A4 + A1^4 * A2 * A3^2 + A1^4 * A4^2 + A3^4 + A1^3 * A3^3$

```

proof -
  let ?x = A3 / A1
  let ?y = (?x ^ 2 + A4) / A1
  have ax: A1 * ?x = A3
    using A1 by simp
  have ay: A1 * ?y = ?x ^ 2 + A4
    using A1 by simp
  have ax2: A1 ^ 2 * ?x ^ 2 = A3 ^ 2
    by (rule scaled-power-eq[OF ax])
  have ax3: A1 ^ 3 * ?x ^ 3 = A3 ^ 3
    by (rule scaled-power-eq[OF ax])
  have ax4: A1 ^ 4 * ?x ^ 4 = A3 ^ 4
    by (rule scaled-power-eq[OF ax])
  have ay2: A1 ^ 2 * ?y ^ 2 = (?x ^ 2 + A4) ^ 2
    by (rule scaled-power-eq[OF ay])
  have square-sum: (?x ^ 2 + A4) ^ 2 = ?x ^ 4 + A4 ^ 2
    using two by (simp add: power2-sum power-mult)
  have yscale:
    A1 ^ 6 * ?y ^ 2 = A3 ^ 4 + A1 ^ 4 * A4 ^ 2
  proof -
    have A1 ^ 6 * ?y ^ 2 = A1 ^ 4 * (A1 ^ 2 * ?y ^ 2)
      by (simp add: power-Suc algebra-simps)
    also have ... = A1 ^ 4 * (?x ^ 2 + A4) ^ 2
      using ay2 by simp
    also have ... = A1 ^ 4 * (?x ^ 4 + A4 ^ 2)
      by (simp only: square-sum)
    also have ... = A3 ^ 4 + A1 ^ 4 * A4 ^ 2
      using ax4 by (simp add: algebra-simps)
    finally show ?thesis .
  qed
  have cross:
    A1 ^ 6 * (A1 * ?x * ?y + A3 * ?y) = 0
  proof -
    have A1 * ?x * ?y + A3 * ?y = A3 * ?y + A3 * ?y
      using ax by simp
    also have ... = 2 * (A3 * ?y)
      by (simp add: algebra-simps)
    also have ... = 0
      using two by simp
    finally show ?thesis
      by simp
  qed
  have xcube:
    A1 ^ 6 * ?x ^ 3 = A1 ^ 3 * A3 ^ 3
  proof -
    have A1 ^ 6 * ?x ^ 3 = A1 ^ 3 * (A1 ^ 3 * ?x ^ 3)
      by (simp add: power-Suc algebra-simps)
    also have ... = A1 ^ 3 * A3 ^ 3
      using ax3 by simp
  proof -

```

```

    finally show ?thesis .
qed
have xsquare:
  
$$A1^6 * (A2 * ?x^2) = A1^4 * A2 * A3^2$$

proof -
  have  $A1^6 * (A2 * ?x^2) =$ 
    
$$A1^4 * A2 * (A1^2 * ?x^2)$$

    by (simp add: power-Suc algebra-simps)
  also have ... =  $A1^4 * A2 * A3^2$ 
    using ax2 by simp
  finally show ?thesis .
qed
have xlinear:
  
$$A1^6 * (A4 * ?x) = A1^5 * A3 * A4$$

proof -
  have  $A1^6 * (A4 * ?x) = A1^5 * A4 * (A1 * ?x)$ 
    by (rule scale-linear-six)
  also have ... =  $A1^5 * A3 * A4$ 
    using ax by (simp add: algebra-simps)
  finally show ?thesis .
qed
have  $A1^6 *$ 
  
$$(((?y^2 + A1 * ?x * ?y + A3 * ?y) -$$

    
$$(?x^3 + A2 * ?x^2 + A4 * ?x + A6))) =$$

  
$$A1^6 * ?y^2 +$$

  
$$A1^6 * (A1 * ?x * ?y + A3 * ?y) -$$

  
$$(A1^6 * ?x^3 + A1^6 * (A2 * ?x^2) +$$

    
$$A1^6 * (A4 * ?x) + A1^6 * A6)$$

  by (simp add: algebra-simps)
also have ... =  $(A3^4 + A1^4 * A4^2) -$ 
  
$$(A1^3 * A3^3 + A1^4 * A2 * A3^2 +$$

    
$$A1^5 * A3 * A4 + A1^6 * A6)$$

  by (simp only: yscale cross xcube xsquare xlinear add-0-right)
also have ... =  $A1^6 * A6 + A1^5 * A3 * A4 +$ 
  
$$A1^4 * A2 * A3^2 + A1^4 * A4^2 +$$

  
$$A3^4 + A1^3 * A3^3$$

  using two
  by (simp add: char-two-neg[OF two] algebra-simps)
finally show ?thesis .
qed

lemma discriminant-char-two-certificate:
  fixes W :: 'a::field weierstrass-coeffs
  assumes two:  $(2 :: 'a) = 0$ 
  and a1:  $a1 \ W \neq 0$ 
  shows  $a1 \ W^6 *$ 
    
$$\text{weierstrass-affine-poly } W \ (a3 \ W / a1 \ W)$$

    
$$(((a3 \ W / a1 \ W)^2 + a4 \ W) / a1 \ W) =$$

    discriminant W

```

```

proof –
  obtain  $A1\ A2\ A3\ A4\ A6$  where  $W$ :
     $W = (a1 = A1, a2 = A2, a3 = A3, a4 = A4, a6 = A6)$ 
  by (cases  $W$ ) blast
  have  $A1: A1 \neq 0$ 
  using  $a1$  unfolding  $W$  by simp
  note  $raw = \text{char-two-affine-certificate-raw}[OF\ two\ A1]$ 
  have polynomial:
     $a1\ W^6 * \text{weierstrass-affine-poly}\ W\ (a3\ W / a1\ W)$ 
     $((a3\ W / a1\ W)^2 + a4\ W) / a1\ W =$ 
     $a1\ W^6 * a6\ W + a1\ W^5 * a3\ W * a4\ W +$ 
     $a1\ W^4 * a2\ W * a3\ W^2 + a1\ W^4 * a4\ W^2 +$ 
     $a3\ W^4 + a1\ W^3 * a3\ W^3$ 
  unfolding  $W$  weierstrass-affine-poly-def
  using  $raw$  by simp
  show ?thesis
  using polynomial discriminant-char-two-expanded[ $OF\ two,\ of\ W$ ]
  by simp
qed

lemma singular-affine-imp-discriminant-zero-char-two:
  fixes  $W :: 'a::\text{field}\ \text{weierstrass-coeffs}$ 
  assumes  $two: (2 :: 'a) = 0$ 
  and singular: singular-affine-point  $W\ x\ y$ 
  shows discriminant  $W = 0$ 
proof (cases  $a1\ W = 0$ )
  case True
  from singular have  $dy: \text{weierstrass-affine-dy}\ W\ x\ y = 0$ 
  by (simp add: singular-affine-point-def)
  have  $a3: a3\ W = 0$ 
  using  $dy\ two\ True$ 
  unfolding weierstrass-affine-dy-def
  by simp
  show ?thesis
  using discriminant-char-two-a1-zero[ $OF\ two\ True$ ]  $a3$  by simp
next
  case False
  from singular have  $F: \text{weierstrass-affine-poly}\ W\ x\ y = 0$ 
  and  $dx: \text{weierstrass-affine-dx}\ W\ x\ y = 0$ 
  and  $dy: \text{weierstrass-affine-dy}\ W\ x\ y = 0$ 
  by (simp-all add: singular-affine-point-def)
  have  $invA1: a1\ W * \text{inverse}\ (a1\ W) = 1$ 
  using False by simp
  have  $dy': a1\ W * x = a3\ W$ 
  using  $dy$ 
  by (simp add: weierstrass-affine-dy-def two add-eq-0-iff2
    char-two-neg[ $OF\ two$ ])
  have  $x: x = a3\ W / a1\ W$ 

```

```

proof (rule eq-divide-imp[OF False])
  show  $x * a1 \ W = a3 \ W$ 
    using  $dy'$  by (simp only: mult.commute)
qed
have  $three: (3 :: 'a) = 1$ 
  using char-two-three[OF two] .
have  $dx'$ :
   $a1 \ W * y - x^2 - a4 \ W = 0$ 
  using  $dx$ 
  by (simp add: weierstrass-affine-dx-def two three)
have  $ay: y * a1 \ W = x^2 + a4 \ W$ 
  by (rule diff-rearrange[OF  $dx'$ ])
have  $y: y = ((a3 \ W / a1 \ W)^2 + a4 \ W) / a1 \ W$ 
  unfolding  $x$ 
proof (rule eq-divide-imp[OF False])
  show  $y * a1 \ W = (a3 \ W / a1 \ W)^2 + a4 \ W$ 
    using  $ay$  unfolding  $x$  .
qed
note  $certificate = discriminant-char-two-certificate$ [OF two False]
show ?thesis
  using  $certificate \ F$ 
  unfolding  $x \ y$ 
  by simp
qed

lemma discriminant-zero-imp-affine-singular-char-two:
  fixes  $W :: 'a::alg-closed-field$   $weierstrass-coeffs$ 
  assumes  $two: (2 :: 'a) = 0$ 
    and  $delta: discriminant \ W = 0$ 
  shows  $\exists x \ y. \text{singular-affine-point } W \ x \ y$ 
proof (cases  $a1 \ W = 0$ )
  case False
  let  $?x = a3 \ W / a1 \ W$ 
  let  $?y = ((a3 \ W / a1 \ W)^2 + a4 \ W) / a1 \ W$ 
  have  $invA1: a1 \ W * inverse \ (a1 \ W) = 1$ 
    using False by simp
  have  $three: (3 :: 'a) = 1$ 
    using char-two-three[OF two] .
  note  $certificate = discriminant-char-two-certificate$ [OF two False]
  have  $F: weierstrass-affine-poly \ W \ ?x \ ?y = 0$ 
    using  $certificate \ delta \ False$  by simp
  have  $ax: a1 \ W * ?x = a3 \ W$ 
    using False by (simp add: mult.commute)
  have  $ay: a1 \ W * ?y = ?x^2 + a4 \ W$ 
    using False by (simp add: mult.commute)
  have  $dx: weierstrass-affine-dx \ W \ ?x \ ?y = 0$ 
    using  $ay$ 
    by (simp add: weierstrass-affine-dx-def two three
      char-two-neg[OF two])

```

```

have dy: weierstrass-affine-dy W ?x ?y = 0
  using ax
  by (simp add: weierstrass-affine-dy-def two char-two-neg[OF two])
show ?thesis
  unfolding singular-affine-point-def
  using F dx dy by blast
next
case True
have a3pow: a3 W ^ 4 = 0
  using discriminant-char-two-a1-zero[OF two True] delta by simp
then have a3: a3 W = 0
  by simp
obtain x where x: x ^ 2 = a4 W
  using nth-root-exists[of 2 a4 W] by auto
obtain y where y:
  y ^ 2 = x ^ 3 + a2 W * x ^ 2 + a4 W * x + a6 W
  using nth-root-exists[of 2
    x ^ 3 + a2 W * x ^ 2 + a4 W * x + a6 W] by auto
have F: weierstrass-affine-poly W x y = 0
  unfolding weierstrass-affine-poly-def
  using True a3 y by simp
have dx: weierstrass-affine-dx W x y = 0
  unfolding weierstrass-affine-dx-def
  using True x
  by (simp add: two char-two-three[OF two] char-two-neg[OF two])
have dy: weierstrass-affine-dy W x y = 0
  unfolding weierstrass-affine-dy-def
  by (simp add: two True a3)
show ?thesis
  unfolding singular-affine-point-def
  using F dx dy by blast
qed

```

12 Odd characteristic

lemma *scale-quadratic*:

```

fixes q x b c :: 'a::comm-ring-1
shows q ^ 2 * (6 * x ^ 2 + b * x + c) =
  6 * (q ^ 2 * x ^ 2) + b * q * (q * x) + c * q ^ 2
by (simp add: power2-eq-square algebra-simps)

```

lemma *c4-h-n-square*:

```

fixes b c d :: 'a::comm-ring-1
shows (18 * d - b * c) ^ 2 =
  324 * d ^ 2 - 36 * b * c * d + b ^ 2 * c ^ 2
by (simp add: power2-eq-square algebra-simps)

```

lemma *c4-h-q-square*:

```

fixes b c :: 'a::comm-ring-1

```

```

shows (b ^ 2 - 24 * c) ^ 2 =
  b ^ 4 - 48 * b ^ 2 * c + 576 * c ^ 2
by (simp add: power2-eq-square power4-eq-xxxx algebra-simps)

lemma c4-h-middle:
  fixes b c d :: 'a::comm-ring-1
  shows b * (b ^ 2 - 24 * c) * (18 * d - b * c) =
    18 * b ^ 3 * d - b ^ 4 * c - 432 * b * c * d +
    24 * b ^ 2 * c ^ 2
  by (simp add: power2-eq-square power3-eq-cube power4-eq-xxxx
    algebra-simps)

lemma c4-h-polynomial:
  fixes b c d :: 'a::comm-ring-1
  shows 6 * (18 * d - b * c) ^ 2 +
    b * (b ^ 2 - 24 * c) * (18 * d - b * c) +
    c * (b ^ 2 - 24 * c) ^ 2 =
    18 * (b ^ 3 * d - b ^ 2 * c ^ 2 - 36 * b * c * d +
    32 * c ^ 3 + 108 * d ^ 2)
  proof -
    note n2 = c4-h-n-square[of d b c]
    note q2 = c4-h-q-square[of b c]
    note middle = c4-h-middle[of b c d]
    show ?thesis
      apply (subst n2)
      apply (subst middle)
      apply (subst q2)
      by (simp add: power2-eq-square power3-eq-cube algebra-simps)
  qed

lemma c4-candidate-h-raw:
  fixes b c d :: 'a::field
  assumes b ^ 2 - 24 * c ≠ 0
  shows (b ^ 2 - 24 * c) ^ 2 *
    (6 * ((18 * d - b * c) / (b ^ 2 - 24 * c)) ^ 2 +
    b * ((18 * d - b * c) / (b ^ 2 - 24 * c)) + c) =
    18 * (b ^ 3 * d - b ^ 2 * c ^ 2 - 36 * b * c * d +
    32 * c ^ 3 + 108 * d ^ 2)
  proof -
    let ?q = b ^ 2 - 24 * c
    let ?n = 18 * d - b * c
    let ?x = ?n / ?q
    have qx: ?q * ?x = ?n
      using assms by simp
    have q2x2: ?q ^ 2 * ?x ^ 2 = ?n ^ 2
      by (rule scaled-power-eq[OF qx])
    have ?q ^ 2 * (6 * ?x ^ 2 + b * ?x + c) =
      6 * (?q ^ 2 * ?x ^ 2) + b * ?q * (?q * ?x) + c * ?q ^ 2
      by (rule scale-quadratic)
  
```

```

also have ... = 6 * ?n ^ 2 + b * ?q * ?n + c * ?q ^ 2
  using qx q2x2 by simp
also have ... = 18 * (b ^ 3 * d - b ^ 2 * c ^ 2 -
  36 * b * c * d + 32 * c ^ 3 + 108 * d ^ 2)
  by (rule c4-h-polynomial)
finally show ?thesis .
qed

lemma c4-candidate-h-certificate:
  fixes W :: 'a::field weierstrass-coeffs
  assumes c4: c4 W ≠ 0
  shows c4 W ^ 2 *
    weierstrass-h W ((18 * b6 W - b2 W * b4 W) / c4 W) =
    -72 * discriminant W
proof -
  note raw = c4-candidate-h-raw[
    OF c4[unfolded c4-def], of b6 W]
  note expanded = four-discriminant-expanded[of W]
  have E:
    b2 W ^ 3 * b6 W - b2 W ^ 2 * b4 W ^ 2 -
    36 * b2 W * b4 W * b6 W + 32 * b4 W ^ 3 +
    108 * b6 W ^ 2 =
    - (4 * discriminant W)
  by (simp only: expanded; simp add: algebra-simps)
show ?thesis
  unfolding c4-def weierstrass-h-def
proof (rule trans)
  show (b2 W ^ 2 - 24 * b4 W) ^ 2 *
    (6 * ((18 * b6 W - b2 W * b4 W) /
      (b2 W ^ 2 - 24 * b4 W)) ^ 2 +
      b2 W * ((18 * b6 W - b2 W * b4 W) /
        (b2 W ^ 2 - 24 * b4 W)) +
      b4 W) =
    18 * (b2 W ^ 3 * b6 W - b2 W ^ 2 * b4 W ^ 2 -
      36 * b2 W * b4 W * b6 W + 32 * b4 W ^ 3 +
      108 * b6 W ^ 2)
  by (rule raw)
  show 18 * (b2 W ^ 3 * b6 W - b2 W ^ 2 * b4 W ^ 2 -
    36 * b2 W * b4 W * b6 W + 32 * b4 W ^ 3 +
    108 * b6 W ^ 2) =
    -72 * discriminant W
  apply (subst E)
  by (simp add: algebra-simps)
qed
qed

```

```

lemma bezout-coefficient-rearrange:
  fixes b c d x :: 'a::comm-ring-1
  shows b ^ 3 + 6 * b ^ 2 * x - 30 * b * c -

```

```

      144 * c * x + 108 * d =
      b ^ 3 - 30 * b * c + 108 * d + 6 * ((b ^ 2 - 24 * c) * x)
proof -
  have x-terms:
    6 * b ^ 2 * x - 144 * c * x = 6 * ((b ^ 2 - 24 * c) * x)
  by (simp only: right-diff-distrib left-diff-distrib;
      simp add: algebra-simps)
  have b ^ 3 + 6 * b ^ 2 * x - 30 * b * c -
    144 * c * x + 108 * d =
    b ^ 3 - 30 * b * c + 108 * d +
    (6 * b ^ 2 * x - 144 * c * x)
  by (simp only: diff-conv-add-uminus; simp add: ac-simps)
  also have ... = b ^ 3 - 30 * b * c + 108 * d +
    6 * ((b ^ 2 - 24 * c) * x)
  by (simp only: x-terms)
  finally show ?thesis .
qed

```

```

lemma c4-candidate-bezout-coefficient:
  fixes W :: 'a::field weierstrass-coeffs
  assumes c4: c4 W ≠ 0
  shows b2 W ^ 3 +
    6 * b2 W ^ 2 * ((18 * b6 W - b2 W * b4 W) / c4 W) -
    30 * b2 W * b4 W -
    144 * b4 W * ((18 * b6 W - b2 W * b4 W) / c4 W) +
    108 * b6 W =
    - c6 W
proof -
  let ?q = b2 W ^ 2 - 24 * b4 W
  let ?n = 18 * b6 W - b2 W * b4 W
  let ?x = ?n / ?q
  have q: ?q ≠ 0
    using c4 unfolding c4-def .
  have qx: ?q * ?x = ?n
    using q by simp
  have b2 W ^ 3 + 6 * b2 W ^ 2 * ?x -
    30 * b2 W * b4 W - 144 * b4 W * ?x + 108 * b6 W =
    b2 W ^ 3 - 30 * b2 W * b4 W + 108 * b6 W +
    6 * (?q * ?x)
  by (rule bezout-coefficient-rearrange)
  also have ... = b2 W ^ 3 - 30 * b2 W * b4 W +
    108 * b6 W + 6 * ?n
  by (simp only: qx)
  also have ... = - c6 W
    unfolding c6-def
  by (simp add: algebra-simps)
  finally show ?thesis
    unfolding c4-def .
qed

```

```

lemma c4-candidate-g-from-h-delta:
  fixes W :: 'a::field weierstrass-coeffs
  assumes c4: c4 W ≠ 0
    and delta: discriminant W = 0
    and h: weierstrass-h W
      ((18 * b6 W - b2 W * b4 W) / c4 W) = 0
  shows weierstrass-g W
    ((18 * b6 W - b2 W * b4 W) / c4 W) = 0
proof -
  let ?x = (18 * b6 W - b2 W * b4 W) / c4 W
  have c6: c6 W ≠ 0
  proof
    assume c6 W = 0
    with c4-c6-discriminant-identity[of W] delta
    have c4 W ^ 3 = 0
    by simp
    with c4 show False
    by simp
  qed
  note coefficient = c4-candidate-bezout-coefficient[OF c4]
  note bezout = four-discriminant-bezout[of W ?x]
  have reduced:
    0 = -
      (b2 W ^ 3 + 6 * b2 W ^ 2 * ?x -
        30 * b2 W * b4 W - 144 * b4 W * ?x + 108 * b6 W) *
      weierstrass-g W ?x
  proof -
    have 0 = 4 * discriminant W
    using delta by simp
    also have ... = -
      (b2 W ^ 3 + 6 * b2 W ^ 2 * ?x -
        30 * b2 W * b4 W - 144 * b4 W * ?x + 108 * b6 W) *
      weierstrass-g W ?x +
      (b2 W ^ 3 * ?x + b2 W ^ 2 * b4 W +
        4 * b2 W ^ 2 * ?x ^ 2 - 28 * b2 W * b4 W * ?x +
        6 * b2 W * b6 W - 32 * b4 W ^ 2 -
        96 * b4 W * ?x ^ 2 + 72 * b6 W * ?x) *
      weierstrass-h W ?x
    by (rule bezout)
    also have ... = -
      (b2 W ^ 3 + 6 * b2 W ^ 2 * ?x -
        30 * b2 W * b4 W - 144 * b4 W * ?x + 108 * b6 W) *
      weierstrass-g W ?x
    by (simp only: h mult-zero-right add-0-right)
    finally show ?thesis .
  qed
  have 0 = c6 W * weierstrass-g W ?x
  using reduced

```

by (simp only: coefficient minus-minus)
 with c6 show ?thesis
 by simp
 qed

lemma square-twelve-scale:
 fixes b x :: 'a::comm-ring-1
 shows $144 * b * x^2 = b * (12 * x)^2$
 by (simp add: power2-eq-square algebra-simps)

lemma zero-h-rearrange:
 fixes x b c :: 'a::comm-ring-1
 shows $24 * (6 * x^2 + b * x + c) =$
 $(12 * x)^2 + 2 * b * (12 * x) + 24 * c$
 by (simp add: power2-eq-square algebra-simps)

lemma zero-g-rearrange:
 fixes x b c d :: 'a::comm-ring-1
 shows $216 * (4 * x^3 + b * x^2 + 2 * c * x + d) =$
 $72 * x^2 * (12 * x + 3 * b) +$
 $36 * c * (12 * x) + 216 * d$
 by (simp add: power2-eq-square power3-eq-cube algebra-simps)

lemma zero-c-invariant-h-raw:
 fixes b c :: 'a::field
 assumes $(12 :: 'a) \neq 0$
 shows $24 * (6 * (-b / 12)^2 + b * (-b / 12) + c) =$
 $-(b^2 - 24 * c)$
 proof -
 let ?x = - b / 12
 have qx: $12 * ?x = - b$
 using assms by simp
 have $24 * (6 * ?x^2 + b * ?x + c) =$
 $(12 * ?x)^2 + 2 * b * (12 * ?x) + 24 * c$
 by (rule zero-h-rearrange)
 also have $\dots = -(b^2 - 24 * c)$
 using qx by (simp add: power2-eq-square algebra-simps)
 finally show ?thesis .
 qed

lemma zero-c-invariant-g-raw:
 fixes b c d :: 'a::field
 assumes $(12 :: 'a) \neq 0$
 shows $216 * (4 * (-b / 12)^3 + b * (-b / 12)^2 +$
 $2 * c * (-b / 12) + d) =$
 $b^3 - 36 * b * c + 216 * d$
 proof -
 let ?x = - b / 12
 have qx: $12 * ?x = - b$

```

    using assms by simp
  have qx2:  $(12 * ?x) ^ 2 = (- b) ^ 2$ 
    using qx by simp
  have 216 *  $(4 * ?x ^ 3 + b * ?x ^ 2 + 2 * c * ?x + d) =$ 
     $72 * ?x ^ 2 * (12 * ?x + 3 * b) +$ 
     $36 * c * (12 * ?x) + 216 * d$ 
    by (rule zero-g-rearrange)
  also have ... =  $144 * b * ?x ^ 2 - 36 * b * c + 216 * d$ 
    using qx by (simp add: algebra-simps)
  also have ... =  $b * (12 * ?x) ^ 2 - 36 * b * c + 216 * d$ 
    by (simp only: square-twelve-scale)
  also have ... =  $b * (- b) ^ 2 - 36 * b * c + 216 * d$ 
    by (simp only: qx2)
  also have ... =  $b ^ 3 - 36 * b * c + 216 * d$ 
    by (simp add: power2-eq-square power3-eq-cube algebra-simps)
  finally show ?thesis .
qed

```

```

lemma zero-c-invariant-h-certificate:
  fixes W :: 'a::field weierstrass-coeffs
  assumes  $(12 :: 'a) \neq 0$ 
  shows  $24 * \text{weierstrass-h } W (- b2 \ W / 12) = - c4 \ W$ 
  using zero-c-invariant-h-raw[OF assms, of b2 W b4 W]
  by (simp add: weierstrass-h-def c4-def)

```

```

lemma zero-c-invariant-g-certificate:
  fixes W :: 'a::field weierstrass-coeffs
  assumes  $(12 :: 'a) \neq 0$ 
  shows  $216 * \text{weierstrass-g } W (- b2 \ W / 12) = - c6 \ W$ 
  using zero-c-invariant-g-raw[OF assms, of b2 W b4 W b6 W]
  by (simp add: weierstrass-g-def c6-def)

```

```

lemma discriminant-zero-imp-affine-singular-odd:
  fixes W :: 'a::alg-closed-field weierstrass-coeffs
  assumes two:  $(2 :: 'a) \neq 0$ 
    and delta:  $\text{discriminant } W = 0$ 
  shows  $\exists x \ y. \text{singular-affine-point } W \ x \ y$ 
proof (cases  $c4 \ W = 0$ )
case False
  let ?x =  $(18 * b6 \ W - b2 \ W * b4 \ W) / c4 \ W$ 
  note h-certificate = c4-candidate-h-certificate[OF False]
  have h:  $\text{weierstrass-h } W \ ?x = 0$ 
    using h-certificate delta False by simp
  have g:  $\text{weierstrass-g } W \ ?x = 0$ 
    by (rule c4-candidate-g-from-h-delta[OF False delta h])
  have singular-affine-point W ?x  $(- (a1 \ W * ?x + a3 \ W) / 2)$ 
    using affine-singular-of-g-h[OF two g h] .
  then show ?thesis
    by blast

```

```

next
case c4zero: True
note invariant-identity = c4-c6-discriminant-identity[of W]
have c6zero:  $c6 \ W = 0$ 
  using invariant-identity c4zero delta by simp
show ?thesis
proof (cases ( $3 :: 'a$ ) = 0)
case three: True
have four: ( $4 :: 'a$ ) = 1
proof -
  have ( $4 :: 'a$ ) =  $3 + 1$ 
  by simp
  also have ... =  $0 + 1$ 
  using three by simp
  also have ... = 1
  by simp
  finally show ?thesis .
qed
have six: ( $6 :: 'a$ ) = 0
proof -
  have ( $6 :: 'a$ ) =  $2 * 3$ 
  by simp
  also have ... = 0
  using three by simp
  finally show ?thesis .
qed
have eight: ( $8 :: 'a$ ) = -1
proof -
  have ( $8 :: 'a$ ) =  $3 * 3 - 1$ 
  by simp
  also have ... =  $0 * 0 - 1$ 
  using three by simp
  also have ... = -1
  by simp
  finally show ?thesis .
qed
have nine: ( $9 :: 'a$ ) = 0
proof -
  have ( $9 :: 'a$ ) =  $3 * 3$ 
  by simp
  also have ... = 0
  using three by simp
  finally show ?thesis .
qed
have twenty-four: ( $24 :: 'a$ ) = 0
proof -
  have ( $24 :: 'a$ ) =  $8 * 3$ 
  by simp
  also have ... = 0

```

```

    using three by simp
    finally show ?thesis .
qed
have twenty-seven:  $(27 :: 'a) = 0$ 
proof -
  have  $(27 :: 'a) = 9 * 3$ 
  by simp
  also have ... = 0
  using three by simp
  finally show ?thesis .
qed
have c4eq:  $b2 \ W^2 - 24 * b4 \ W = 0$ 
  using c4zero unfolding c4-def .
have b2pow:  $b2 \ W^2 = 0$ 
  using c4eq twenty-four by simp
then have b2zero:  $b2 \ W = 0$ 
  by simp
have deltaeq:

$$-(b2 \ W^2) * b8 \ W - 8 * b4 \ W^3 - 27 * b6 \ W^2 + 9 * b2 \ W * b4 \ W * b6 \ W = 0$$

  using delta unfolding discriminant-def .
have b4pow:  $b4 \ W^3 = 0$ 
  using deltaeq b2zero eight nine twenty-seven
  by simp
then have b4zero:  $b4 \ W = 0$ 
  by simp
obtain x where  $x: x^3 = - b6 \ W$ 
  using nth-root-exists[of 3 - b6 W] by auto
have g:  $weierstrass-g \ W \ x = 0$ 
  unfolding weierstrass-g-def
  using four b2zero b4zero x by simp
have h:  $weierstrass-h \ W \ x = 0$ 
  unfolding weierstrass-h-def
  using six b2zero b4zero by simp
have singular-affine-point  $W \ x \ (- (a1 \ W * x + a3 \ W) / 2)$ 
  using affine-singular-of-g-h[OF two g h] .
then show ?thesis
  by blast
next
case three: False
have twelve-as-product:  $(12 :: 'a) = 2 * 2 * 3$ 
  by simp
have twelve:  $(12 :: 'a) \neq 0$ 
proof
  assume twelve-zero:  $(12 :: 'a) = 0$ 
  have product-zero:  $(2 :: 'a) * 2 * 3 = 0$ 
  proof -
    have  $(2 :: 'a) * 2 * 3 = 12$ 
    by simp

```

```

    also have ... = 0
      by (rule twelve-zero)
    finally show ?thesis .
  qed
  have product-nonzero: (2 :: 'a) * 2 * 3 ≠ 0
  proof
    assume product-zero: (2 :: 'a) * 2 * 3 = 0
    have ((2 :: 'a) = 0 ∨ (2 :: 'a) = 0) ∨ (3 :: 'a) = 0
      using product-zero by (simp only: mult-eq-0-iff)
    with two three show False
      by blast
  qed
  with product-zero show False
    by contradiction
  qed
  have inv12: (12 :: 'a) * inverse 12 = 1
    using twelve by simp
  let ?x = - b2 W / 12
  have twenty-four-as-product: (24 :: 'a) = 2 * 12
    by simp
  have twenty-four: (24 :: 'a) ≠ 0
  proof
    assume twenty-four-zero: (24 :: 'a) = 0
    have product-zero: (2 :: 'a) * 12 = 0
    proof -
      have (2 :: 'a) * 12 = 24
        by simp
      also have ... = 0
        by (rule twenty-four-zero)
      finally show ?thesis .
    qed
  qed
  have product-nonzero: (2 :: 'a) * 12 ≠ 0
  proof
    assume product-zero: (2 :: 'a) * 12 = 0
    have (2 :: 'a) = 0 ∨ (12 :: 'a) = 0
      using product-zero by (simp only: mult-eq-0-iff)
    with two twelve show False
      by blast
  qed
  with product-zero show False
    by contradiction
  qed
  have two-hundred-sixteen-as-product:
    (216 :: 'a) = 2 * 3 * 3 * 12
    by simp
  have two-hundred-sixteen: (216 :: 'a) ≠ 0
  proof
    assume two-hundred-sixteen-zero: (216 :: 'a) = 0
    have product-zero: (2 :: 'a) * 3 * 3 * 12 = 0

```

```

proof –
  have  $(2 :: 'a) * 3 * 3 * 12 = 216$ 
    by simp
  also have  $\dots = 0$ 
    by (rule two-hundred-sixteen-zero)
  finally show ?thesis .
qed
have product-nonzero:  $(2 :: 'a) * 3 * 3 * 12 \neq 0$ 
proof
  assume product-zero:  $(2 :: 'a) * 3 * 3 * 12 = 0$ 
  have  $((2 :: 'a) = 0 \vee (3 :: 'a) = 0) \vee$ 
     $(3 :: 'a) = 0) \vee (12 :: 'a) = 0$ 
    using product-zero by (simp only: mult-eq-0-iff)
  with two three twelve show False
    by blast
qed
with product-zero show False
  by contradiction
qed
note g-certificate = zero-c-invariant-g-certificate[OF twelve, of W]
note h-certificate = zero-c-invariant-h-certificate[OF twelve, of W]
have g: weierstrass-g W ?x = 0
  using g-certificate c6zero two-hundred-sixteen by simp
have h: weierstrass-h W ?x = 0
  using h-certificate c4zero twenty-four by simp
have singular-affine-point W ?x  $(- (a1 \ W * ?x + a3 \ W) / 2)$ 
  using affine-singular-of-g-h[OF two g h] .
then show ?thesis
  by blast
qed
qed

```

13 The discriminant criterion

```

lemma singular-affine-imp-discriminant-zero:
  fixes W :: 'a::field weierstrass-coeffs
  assumes singular-affine-point W x y
  shows discriminant W = 0
proof (cases  $(2 :: 'a) = 0$ )
  case True
  show ?thesis
    using singular-affine-imp-discriminant-zero-char-two[OF True assms] .
next
  case False
  show ?thesis
    using singular-affine-imp-discriminant-zero-odd[OF False assms] .
qed

theorem projective-singular-iff-discriminant-zero:

```

```

fixes  $W :: 'a::alg-closed-field\ weierstrass-coeffs$ 
shows  $(\exists p. \text{singular-projective-point } W\ p) \longleftrightarrow$ 
 $\text{discriminant } W = 0$ 
proof -
  have affine:
     $(\exists x\ y. \text{singular-affine-point } W\ x\ y) \longleftrightarrow$ 
 $\text{discriminant } W = 0$ 
  proof
    assume  $\exists x\ y. \text{singular-affine-point } W\ x\ y$ 
    then show  $\text{discriminant } W = 0$ 
      using singular-affine-imp-discriminant-zero by blast
  next
    assume delta:  $\text{discriminant } W = 0$ 
    show  $\exists x\ y. \text{singular-affine-point } W\ x\ y$ 
    proof (cases  $(2 :: 'a) = 0$ )
      case True
        show ?thesis
          using discriminant-zero-imp-affine-singular-char-two[OF True delta] .
      next
        case False
          show ?thesis
            using discriminant-zero-imp-affine-singular-odd[OF False delta] .
    qed
  qed
  show ?thesis
    using projective-singular-iff-affine-singular[of W] affine by blast
qed

lemma b2-map-to-ac [simp]:
 $b2\ (\text{map-weierstrass-coeffs } to-ac\ W) = to-ac\ (b2\ W)$ 
by (simp add: b2-def)

lemma b4-map-to-ac [simp]:
 $b4\ (\text{map-weierstrass-coeffs } to-ac\ W) = to-ac\ (b4\ W)$ 
by (simp add: b4-def)

lemma b6-map-to-ac [simp]:
 $b6\ (\text{map-weierstrass-coeffs } to-ac\ W) = to-ac\ (b6\ W)$ 
by (simp add: b6-def)

lemma b8-map-to-ac [simp]:
 $b8\ (\text{map-weierstrass-coeffs } to-ac\ W) = to-ac\ (b8\ W)$ 
by (simp add: b8-def)

lemma c4-map-to-ac [simp]:
 $c4\ (\text{map-weierstrass-coeffs } to-ac\ W) = to-ac\ (c4\ W)$ 
by (simp add: c4-def)

lemma c6-map-to-ac [simp]:

```

```

c6 (map-weierstrass-coeffs to-ac W) = to-ac (c6 W)
by (simp add: c6-def)

lemma discriminant-map-to-ac [simp]:
  discriminant (map-weierstrass-coeffs to-ac W) =
    to-ac (discriminant W)
by (simp add: discriminant-def)

definition projective-weierstrass-nonsingular ::
  'a::field weierstrass-coeffs  $\Rightarrow$  bool
where
  projective-weierstrass-nonsingular W  $\longleftrightarrow$ 
    projective-weierstrass-nonsingular-over
      (map-weierstrass-coeffs to-ac W)

theorem weierstrass-nonsingular-iff:
  fixes W :: 'a::field weierstrass-coeffs
  shows projective-weierstrass-nonsingular W  $\longleftrightarrow$ 
    discriminant W  $\neq$  0
  unfolding projective-weierstrass-nonsingular-def
    projective-weierstrass-nonsingular-over-def
  using projective-singular-iff-discriminant-zero[
    of map-weierstrass-coeffs to-ac W]
  by simp

end

theory Weierstrass-j-Invariant
  imports Weierstrass-Discriminant-Criterion
begin

```

14 The partial j-invariant

```

definition weierstrass-j ::
  'a::field weierstrass-coeffs  $\Rightarrow$  'a option
where
  weierstrass-j W =
    (if projective-weierstrass-nonsingular W
      then Some (c4 W ^ 3 / discriminant W)
      else None)

lemma weierstrass-j-nonsingular:
  assumes projective-weierstrass-nonsingular W
  shows weierstrass-j W = Some (c4 W ^ 3 / discriminant W)
  using assms by (simp add: weierstrass-j-def)

lemma weierstrass-j-discriminant-nonzero:
  assumes discriminant W  $\neq$  0
  shows weierstrass-j W = Some (c4 W ^ 3 / discriminant W)
  using assms

```

by (simp add: weierstrass-j-def weierstrass-nonsingular-iff)

lemma *weierstrass-j-singular*:
 assumes $\neg$ projective-weierstrass-nonsingular *W*
 shows *weierstrass-j W = None*
 using assms by (simp add: weierstrass-j-def)

lemma *weierstrass-j-eq-none-iff* [simp]:
 $\text{weierstrass-j } W = \text{None} \longleftrightarrow \text{discriminant } W = 0$
 unfolding *weierstrass-j-def*
 using *weierstrass-nonsingular-iff*[of *W*]
 by auto

lemma *weierstrass-j-eq-some-iff*:
 $\text{weierstrass-j } W = \text{Some } j \longleftrightarrow$
 $\text{discriminant } W \neq 0 \wedge j = c_4 \, W^{\wedge} 3 \, / \, \text{discriminant } W$
 unfolding *weierstrass-j-def*
 using *weierstrass-nonsingular-iff*[of *W*]
 by auto

lemma *weierstrass-j-defined-iff*:
 $\text{weierstrass-j } W \neq \text{None} \longleftrightarrow$
 projective-weierstrass-nonsingular *W*
 unfolding *weierstrass-j-def*
 by auto

end
theory *Short-Weierstrass-Bridge*
 imports
 Weierstrass-j-Invariant
 Elliptic-Curves-Group-Law.Elliptic-Axclass
 begin

15 Specialization to short Weierstrass equations

lemma *discriminant-short-weierstrass*:
 fixes *A B* :: 'a::comm-ring-1
 shows $\text{discriminant } (\text{short-weierstrass-coeffs } A \, B) =$
 $-16 * (4 * A^{\wedge} 3 + 27 * B^{\wedge} 2)$
 unfolding *discriminant-def b2-def b4-def b6-def b8-def*
 short-weierstrass-coeffs-def
 by (simp add: algebra-simps power2-eq-square power3-eq-cube)

lemma *on-weierstrass-affine-short*:
 fixes *A B x y* :: 'a::comm-ring-1
 shows $\text{on-weierstrass-affine } (\text{short-weierstrass-coeffs } A \, B) \, x \, y$
 $\longleftrightarrow y^{\wedge} 2 = x^{\wedge} 3 + A * x + B$
 unfolding *on-weierstrass-affine-def short-weierstrass-coeffs-def*
 by simp

lemma *on-weierstrass-affine-short-iff-on-curve*:

fixes $A\ B\ x\ y :: 'a::\text{ell-field}$

shows *on-weierstrass-affine* (*short-weierstrass-coeffs* $A\ B$) $x\ y$

$\longleftrightarrow$ *on-curve* $A\ B$ (*Point* $x\ y$)

unfolding *on-curve-def*

by (*simp add: on-weierstrass-affine-short*)

theorem *projective-weierstrass-nonsingular-short-iff*:

fixes $A\ B :: 'a::\text{ell-field}$

shows *projective-weierstrass-nonsingular*

(*short-weierstrass-coeffs* $A\ B$)

$\longleftrightarrow$ *nonsingular* $A\ B$

proof –

have *two*: $(2 :: 'a) \neq 0$

by (*rule two-not-zero*)

have *four*: $(4 :: 'a) \neq 0$

by (*rule four-ne-zero-of-two-ne-zero[OF two]*)

have *sixteen*: $(16 :: 'a) \neq 0$

proof

assume *sixteen-zero*: $(16 :: 'a) = 0$

have *product-zero*: $(4 :: 'a) * 4 = 0$

proof –

have $(4 :: 'a) * 4 = 16$

by *simp*

also have $\dots = 0$

by (*rule sixteen-zero*)

finally show *?thesis* .

qed

have $(4 :: 'a) = 0 \vee (4 :: 'a) \neq 0$

using *product-zero* **by** (*simp only: mult-eq-0-iff*)

with four show *False*

by *blast*

qed

have *neg-sixteen*: $(-16 :: 'a) \neq 0$

using *sixteen* **by** *simp*

have *delta-iff*:

discriminant (*short-weierstrass-coeffs* $A\ B$) $\neq 0$

$\longleftrightarrow 4 * A^3 + 27 * B^2 \neq 0$

unfolding *discriminant-short-weierstrass*

using *neg-sixteen*

by (*simp only: mult-eq-0-iff; blast*)

show *?thesis*

unfolding *nonsingular-def*

using *weierstrass-nonsingular-iff*[

of short-weierstrass-coeffs $A\ B$] *delta-iff*

by *blast*

qed

end

References

- [1] S. Berghofer. The group law for elliptic curves. *Archive of Formal Proofs*, February 2017. https://isa-afp.org/entries/Elliptic_Curves_Group_Law.html, Formal proof development.
- [2] J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, second edition, 2009. DOI: <https://doi.org/10.1007/978-0-387-09494-6>.