A Computationally Surveyable Proof of the Group Properties of an Elliptic Curve

David M. Russinoff
david@russinoff.com

We present an elementary proof of the abelian group properties of the elliptic curve known as *Curve25519*, as a component of a comprehensive proof of correctness of a hardware implementation of the associated Diffie-Hellman key agreement algorithm. The entire proof has been formalized and mechanically verified with ACL2, and is *computationally surveyable* in the sense that all steps that require mechanical support are presented in such a way that they may be readily reproduced in any suitable programming language.

1 Introduction

An effort is under way at Intel to develop and verify a formal model and a hardware implementation of the elliptic curve key agreement algorithm known as *Curve25519* [2], using ACL2. The most challenging aspect of this problem is the proof of the abelian group properties of the curve, especially associativity. This result may be viewed either as a deep theorem of algebraic or projective geometry [6, 3], accessible only to experts in that field, or as an elementary but computationally intensive arithmetic exercise, involving, as Bernstein [2] observes, "standard (but lengthy) calculations". Silverman and Tate [10] attempt to quantify the effort with a "tongue-in-cheek estimate":

> Of course, there are a lot of cases to consider .... But in a few days you will be able to check associativity using these formulas. So we need say nothing more about the proof of the associative law!

What remains to be said is that there is compelling evidence (see below) that an elementary hand proof of this result is a practical impossibility. The first serious attack on the problem, by Friedl [4], was a combination of mathematical analysis and symbolic computation with the CoCoA (Computations in Commutative Algebra) package. Building on Friedl's results, Théry [11] later developed a comprehensive formal proof with Coq. (Both papers address the somewhat more general class of Weierstrass curves rather than the one on which we focus here, but there is no difference in computational complexity.) These two efforts, which together represent a significant achievement, may be contrasted in the terminology of automated reasoning [1]: Friedl's work is *accepting* insofar as it treats CoCoA as a trusted oracle, whereas Théry's proof is *autarkic* by virtue of performing all logical deductions and supporting computations within the same formal system.

From a traditional mathematical perspective, however, both of these results are open to the same common criticism of computer-assisted proofs. There is general agreement in the mathematical community that it is desirable for a proof to be *surveyable* in the sense that each of its assertions could be derived manually by a competent reader as a logical consequence of preceding assertions and otherwise established results, and that the proof is short enough to be comprehended. One goal behind this principle is correctness, but equally important is the desire for mathematical growth—the propagation and advancement of techniques and ideas.

Warren Hunt Jr. and Anna Slobodova (Eds.):
ACL2 Theorem Prover and its Applications
EPTCS 249, 2017, pp. 30–46, doi:10.4204/EPTCS.249.3

In the realm of computing, this is often an unattainable objective—reliance on a mechanical proof assistant may be unavoidable. It is common to find in a published proof in this field, in lieu of a cogent argument, an appeal to the authority of an established proof system. This device is a stark realization of Tymoczko's allegory of the infallible Martian genius whose proclamations go unquestioned—proof by "Simon Says" [12]. It may provide evidence of correctness but does little to illuminate the underlying mathematics.

Dependence on mechanical assistance, however, need not preclude a full exposition of a proof. For example, the correctness of a hardware divider typically depends on a relation between the value and indices of each entry of an array that is too large to be either generated or checked by hand, but it should be possible to characterize the computation in such a way that it can be understood and machine-checked by the skeptical reader.

This suggests a judicious weakening of the conventional notion of surveyability. A proof may be said to be *computationally surveyable* if its only departure from that notion is its dependence on unproved assertions that satisfy the following criteria:

(1) The assertion pertains to a function for which a clear constructive definition has been provided, and merely specifies the value of that function corresponding to a concrete set of arguments.

(2) The computation of this value has been performed mechanically by the author of the proof in a reasonably short time.

(3) A competent reader could readily code the function in the programming language of his choice and verify the asserted result on his own computing platform.

Such a proof, though still objectionable to those who insist on strict surveyability, can convey a comprehensive understanding of a theorem and is susceptible to a process of social review, thus oppugning a commonly stated basis for the objection.

Neither of the treatments of the elliptic curve group properties cited above attempts such a proof, perhaps because the supporting tool or its application to the problem at hand is too complicated to admit a concise specification. Thus, Friedl simply attributes unproved results to CoCoA, while Théry's claims depend on an undisclosed "tactic" that has reportedly been implemented in Coq.

An integrated computationally surveyable proof of a result of this sort, which combines subtle mathematical analysis with intensive computation, is best carried out with the support of an interactive prover based on an efficient executable logic. We shall present such a proof of this theorem that has been formalized and mechanically checked in its entirety with ACL2. The computational results for which we rely solely on ACL2 for verification, as opposed to proof checking, (all of which are in Section 7) are labeled as **Computation**s, and are thus clearly distinguished from other steps in the proof, which are listed as **Lemma**s. All computations are performed on S-expressions and are most naturally performed in LISP, but can be readily implemented in any language that provides linked lists. Moreover, our exposition is confined to conventional mathematical terminology and notation, with no reference to the ACL2 logic.

Our proof benefits significantly from the two earlier efforts, both in its overall approach and through its appropriation of specific lemmas. In particular, we follow [11] in the representation of polynomials in sparse Horner normal form, using a normalization procedure adapted from [5]. Furthermore, our Lemmas 2.1, 2.2, and 7.7 are variants of results found in [11] (two of which are inherited from [4]).

The supporting materials for this paper include several subdirectories of `books/projects` in the ACL2 repository. The main script resides in `curve25519`. The basis of Curve25519 is the primality of $\mathscr{p} = 2^{255} - 19$, which is proved in `quadratic-reciprocity` by Pratt's method [7]and explained in [9]. Fermat's Theorem ($a^{\mathscr{p}-1} \bmod \mathscr{p} = 1$ when a is not divisible by $\mathscr{p}$), which allows the inversion operator in the field $\mathbb{F}_{\mathscr{p}}$ to be defined as $a^{-1} = a^{\mathscr{p}-2} \bmod \mathscr{p}$, is also formalized in `quadratic-reciprocity`.

Our formalization of sparse Horner normal forms is in the subdirectory `shnf`. Following [5], we define an efficiently computable normalization of polynomial terms and an evaluation function on normal forms, and prove equality between the value of a polynomial and that of its representation, for all variable assignments. Thus, the equivalence of two polynomials we may be established by computing their normalizations and observing that they coincide.

Of course, the utility of this method rests on the property of completeness: equivalent polynomials always produce the same representation. According to the authors of the Coq proof, which does not address this property, it cannot even be stated within their formal framework. Our development includes a constructive proof of this result that we have formalized in ACL2, based on a function that computes, for a given pair of two polynomials, a list of variable assignments for which the values of the polynomials differ, whenever such a list exists. This result is not required for our present purpose, but is documented elsewhere [8].

The distinguishing features of our proof that enable the objective of computational surveyability are (1) a specialized rewriting procedure that reduces the normal form of a polynomial according to the curve equation (Definition 5.5), and (2) an encoding of group elements as integer triples, which facilitates symbolic computation of the group operation (Definition 6.3). Both of these functions require automated computation but admit concise specifications and correctness proofs. Furthermore, a modest improvement in efficiency over the more general Coq proof tactic is suggested by a comparison of execution times of the three computational results of [11] (9.2, 3.9, and 18.8 seconds for `spec3_assoc`, `spec2_assoc`, and `spec3_assoc`, respectively) and our versions of the same computations (3.78, 0.36, and 3.8 seconds for Computations 7, 8, and 9). We exploit this facility by performing several more intensive computations, thereby eliminating much of Théry's analysis, which he characterizes as "really tedious". In particular, Computation 10, which is proved in 26.2 seconds, disposes of a critical case of associativity. It is also worth noting that if the polynomial involved in this result were expanded into a sum of monomials, as might be done in a direct hand proof based on "standard computations", the number of terms would exceed 10^{25}. Clearly, the reader who completes such a proof "in a few days" is exceptionally good with figures.

2 Curve25519

Let $\wp = 2^{255} - 19$ and $A = 486662$. The primality of $\wp$ is proved in [9]. The field of order $\wp$ is the set $\mathbb{F}_\wp = \{0, 1, 2, \ldots, \wp - 1\}$ with the operations of addition and multiplication modulo $\wp$. Every $n \in \mathbb{Z}$ naturally corresponds to the field element n mod $\wp$, which we denote as $\bar{n}$. The field operations will be denoted by the usual symbols: if $x \in \mathbb{F}_\wp$, $y \in \mathbb{F}_\wp$, and $k \in \mathbb{N}$, then "$x+y$", "$x-y$", "$-x$", "xy", or "x^k" may refer to an operation in either $\mathbb{F}_\wp$ or $\mathbb{Z}$, depending on context, whereas "x/y" will only denote an operation in $\mathbb{F}_\wp$.

Definition 2.1 $EC = \{(x,y) \in \mathbb{F}_\wp \times \mathbb{F}_\wp \mid y^2 = x^3 + Ax^2 + x\} \cup \{\infty\}$.

Our goal is to show that EC is an abelian group under the following operation:

Definition 2.2 *Let $P \in EC$ and $Q \in EC$.*

(1) $P \oplus \infty = \infty \oplus P = P$.

(2) If $P = (x,y)$, then $P \oplus (x,-y) = \infty$.

(3) If $P = (x_1,y_1)$, $Q = (x_2,y_2) \neq (x_1,-y_1)$, and $\lambda = \begin{cases} \frac{y_2-y_1}{x_2-x_1} & \text{if } x_1 \neq x_2 \\ \frac{3x_1^2+2Ax_1+1}{2y_1} & \text{if } x_1 = x_2, \end{cases}$ *then $P \oplus Q = (x,y)$, where $x = \lambda^2 - A - x_1 - x_2$ and $y = \lambda(x_1 - x) - y_1$.*

Clearly, ∞ is the identity element, the inverse of $P = (x,y)$ is $\ominus P = (x,-y)$, and according to Corollary 1 of [9], the origin $O = (0,0)$ is the only element of order 2.

Remark. If we consider Definition 2.1 as an equation over $\mathbb{R}$ instead of $\mathbb{F}_{\wp}$, then Definition 2.2 admits a simple geometric interpretation. Except when $Q = \ominus P$, the line connecting points P and Q on the curve (or the tangent line at P, in case $P = Q$) intersects the curve at another point, R. If we were to define the operation as $P \oplus Q = \ominus R$, then analytic geometry would yield the formula in the definition. If $Q = \ominus P$, the third point of intersection is taken to be ∞.

In the sequel, we shall assume that $P_0 = (x_0,y_0)$, $P_1 = (x_1,y_1)$, and $P_2 = (x_2,y_2)$ are fixed elements of EC that are distinct from ∞ (but not necessarily from one another), in order to obviate repetition of such hypotheses. Any result pertaining to these points may be generalized by replacing them with arbitrary finite points of EC. We begin with two simple consequences of Definition 2.2.

Lemma 2.1 $P_0 \oplus P_1 \neq P_0$.

PROOF: Suppose $P_0 \oplus P_1 = P_0$. Equating y-coordinates, we have $y_0 = \lambda(x_0 - x_0) - y_0 = -y_0$, which implies $2y_0 = 0$ and hence (since $\mathbb{F}_{\wp}$ is of odd characteristic $\wp$) $y_0 = 0$, which implies $x_0 = 0$. But x_1 cannot be 0, as this would imply $P_0 = O \oplus O = \infty$. Thus, the equation $x_0 = \lambda^2 - A - x_0 - x_1$ reduces to

$$\frac{y_1^2}{x_1^2} = \lambda^2 = x_1 + A,$$

which implies $x_1^3 + Ax_1^2 + x_1 = y_1^2 = x_1^3 + Ax_1^2$, contradicting $x_1 \neq 0$. □

Lemma 2.2 *If $P_0 \oplus P_1 = P_0 \oplus (\ominus P_1)$, then either $P_0 = O$ or $P_1 = O$.*

PROOF: If $P_0 = \ominus P_1$, then

$$P_0 \oplus P_0 = P_0 \oplus (\ominus P_1) = P_0 \oplus P_1 = \ominus P_1 \oplus P_1 = \infty,$$

which implies $P_0 = O$. Similarly, if $P_0 = P_1$, then

$$P_0 \oplus P_0 = P_0 \oplus P_1 = P_0 \oplus (\ominus P_1) = P_0 \oplus (\ominus P_0) = \infty.$$

Therefore, we may assume $x_0 \neq x_1$. Equating the x-coordinates of $P_0 \oplus P_1$ and $P_0 \oplus (\ominus P_1)$, we have

$$\left(\frac{y_1 - y_0}{x_1 - x_0}\right)^2 - x_0 - x_1 = \left(\frac{y_1 + y_0}{x_1 - x_0}\right)^2 - x_0 - x_1,$$

which implies $4y_0y_1 = 0$, and hence either $y_0 = 0$ or $y_1 = 0$. □

3 Encoding Points on the Curve as Integer Triples

Our scheme for symbolic computation of the group operation is based on a mapping from $\mathbb{Z}^3$ to $\mathbb{F}_{\wp}^2$:

Definition 3.1 *If $\mathscr{P} = (m,n,z) \in \mathbb{Z}^3$, where z is not divisible by $\wp$, then*

$$decode(\mathscr{P}) = \left(\frac{\bar{m}}{\bar{z}^2}, \frac{\bar{n}}{\bar{z}^3}\right) \in \mathbb{F}_{\wp}^2$$

and $\mathscr{P}$ is said to be an encoding of $decode(\mathscr{P})$.

Note that every $P=(x,y)\in\mathbb{F}_{\wp}^2$ admits the canonical encoding $\mathscr{P}=(x,y,1)$.

The motivation for this definition is that an encoding of $P\oplus Q$ can often be readily derived from encodings of P and Q in certain cases of interest. We define a partial addition operation on $\mathbb{Z}^3$ corresponding to Definition 2.2:

Definition 3.2 *Given $\mathscr{P}\in\mathbb{Z}^3$ and $\mathscr{Q}\in\mathbb{Z}^3$, $\mathscr{P}\oplus\mathscr{Q}\in\mathbb{Z}^3$ is defined in two cases:*

(1) If $\mathscr{P}=\mathscr{Q}=(m,n,z)$, then $\mathscr{P}\oplus\mathscr{Q}=(m',n',z')$, where

$$\begin{aligned} z' &= z_{dbl}(\mathscr{P})=2nz,\\ w' &= w_{dbl}(\mathscr{P})=3m^2+2Amz^2+z^4,\\ m' &= m_{dbl}(\mathscr{P})=w'^2-4n^2(Az^2+2m),\\ n' &= n_{dbl}(\mathscr{P})=w'(4mn^2-m')-8n^4. \end{aligned}$$

(2) If $\mathscr{P}=(x,y,1)\in\mathbb{Z}^3$ and $\mathscr{Q}=(m,n,z)\neq\mathscr{P}$, then $\mathscr{P}\oplus\mathscr{Q}=(m',n',z')$, where

$$\begin{aligned} z' &= z_{sum}(\mathscr{P},\mathscr{Q})=z(z^2x-m),\\ m' &= m_{sum}(\mathscr{P},\mathscr{Q})=\left(z^3y-n\right)^2-\left(z^2(A+x)+m\right)\left(z^2x-m\right)^2\\ n' &= n_{sum}(\mathscr{P},\mathscr{Q})=\left(z^3y-n\right)\left(z'^2x-m'\right)-z'^3y. \end{aligned}$$

Lemma 3.1 *Let $P=decode(\mathscr{P})\in EC$ and $Q=decode(\mathscr{Q})\in EC$, where $\mathscr{P}\oplus\mathscr{Q}$ is defined and if $P=Q$, then $P\neq O$ and $\mathscr{P}=\mathscr{Q}$. Then $decode(\mathscr{P}\oplus\mathscr{Q})=P\oplus Q$.*

PROOF: The arithmetic operations below are to be understood as operations in $\mathbb{F}_{\wp}$ on the field elements corresponding to the integers involved.

We first consider the case $\mathscr{P}=\mathscr{Q}=(m,n,z)$. Let

$$\lambda=\frac{3\left(\frac{m}{z^2}\right)^2+2A\left(\frac{m}{z^2}\right)+1}{2\left(\frac{n}{z^3}\right)}=\frac{3m^2+2Amz^2+z^4}{2nz}=\frac{w'}{z'}.$$

Then $P\oplus P=(x,y)$, where

$$x=\lambda^2-A-2\left(\frac{m}{z^2}\right)=\frac{w'^2}{z'^2}-\frac{Az^2+2m}{z^2}=\frac{w'^2}{z'^2}-\frac{4n^2(Az^2+2m)}{z'^2}=\frac{m'}{z'^2}$$

and

$$y=\lambda\left(\frac{m}{z^2}-x\right)-\frac{n}{z^3}=\frac{w'}{z'}\cdot\frac{4mn^2-m'}{z'^2}-\frac{8n^4}{z'^3}=\frac{w'(4mn^2-m')-8n^4}{z'^3}=\frac{n'}{z'^3}.$$

Thus, $decode(\mathscr{P}\oplus\mathscr{P})=decode(m',n',z')=(x,y)=P\oplus P$.

In the remaining case, we have $\mathscr{P}=(m,n,z)$ and $\mathscr{Q}=(x,y,1)$. Let

$$\lambda=\frac{y-\frac{n}{z^3}}{x-\frac{m}{z^2}}=\frac{z^3y-n}{z(z^2x-m)}=\frac{z^3y-n}{z'},$$

Then $P\oplus Q=(x',y')$, where

$$\begin{aligned} x' &= \lambda^2-A-x-\left(\frac{m}{z^2}\right)=\frac{(z^3y-x)^2}{z'^2}-\frac{z^2(A+x)+m}{z^2}\\ &= \frac{(z^3y-x)^2}{z'^2}-\frac{\left(z^2(A+x)+m\right)\left(z^2x-m\right)^2}{z'^2}=\frac{m'}{z'^2} \end{aligned}$$

and

$$y' = \lambda(x - x') - y = \frac{z^3 y - n}{z'} \cdot \frac{xz'^2 - m'}{z'^2} - \frac{z'^3 y}{z'^3} = \frac{n'}{z'^3}.$$

Thus, $decode(\mathscr{P} \oplus \mathscr{Q}) = decode(m', n', z') = (x', y') = P \oplus Q$. □

4 Polynomial Terms and Sparse Horner Normal Form

In this section, we describe our formalization of sparse Horner forms as S-expressions. For this purpose, an S-expression is an integer, a symbol, or an ordered list $s = (s_0\ s_1\ \ldots\ s_n)$ of S-expressions. In the last case, $head(s) = s_0$, and for $k \in \mathbb{N}$, we define $s^{(k)} = (s_k\ \ldots\ s_n)$. The set of all lists whose members are confined to a set S is $\mathscr{L}(S)$.

Under the usual ACL2 encoding of multi-variable polynomials, a *polynomial term* over a list V of variable symbols is an S-expression constructed from integers and symbols in V using the symbols `+`, `-`, `*`, and `EXPT`. The function *evalp* evaluates a term according to an alist that associates variables with integer values in the natural way. For example, if $V =$ `(X Y Z)` and $A =$ `((X 2) (Y 3) (Z 0))`, then $\tau =$ `(* X (EXPT (+ Y Z) 3))` is a term over V and $evalp(\tau, A) = 2 \cdot (3+0)^3 = 54$. The set of all polynomial terms over V is denoted $\mathscr{T}(V)$.

We shall represent polynomial terms as objects of the following type:

Definition 4.1 *A sparse Horner form (SHF) is any of the following:*

(a) An integer;

(b) A list `(POP i p)`*, where* $i \in \mathbb{N}$ *and* p *is a SHF*

(c) A list `(POW i p q)`*, where* $i \in \mathbb{N}$ *and* p *and* q *are SHFs.*

A SHF is normal if its components are normal and it is not either of the following:

(a) `(POP i p)`*, where* $i = 0$ *or* $p \in \mathbb{Z}$ *or* $p =$ `(POP j q)`*;*

(b) `(POW i p q)`*, where* $i = 0$ *or* $p =$ `(POW j r 0)`*.*

$\mathscr{H}$ *denotes the set of all normal SHFs, or SHNFs.*

The evaluation of a SHF with respect to a list of integers is defined as follows:

Definition 4.2 *Let* h *be a SHF and let* $N \in \mathscr{L}(Z)$*.*

(a) If $h \in \mathbb{Z}$*, then* $evalh(h, N) = h$*.*

(b) If $h =$ `(POP i p)`*, then then* $evalh(h, N) = evalh(p, N^{(i)})$*.*

(c) If $h =$ `(POW i p q)` *and* $head(N) = n$*, then* $evalh(h, N) = n^i evalh(p, N) + evalh(q, N^{(1)})$*.*

(d) If $h =$ `(POW i p q)` *and* $N =$ `()`*, then* $evalh(h, N) = 0$*.*

Our objective is to define, for a given variable list $V =$ `(`$v_0 \ldots v_k$`)`, a mapping $norm : \mathscr{T}(V) \to \mathscr{H}$ such that if $N =$ `(`$n_0 \ldots n_k$`)` and $A =$ `((`v_0 n_0`)...(`v_k n_k`))`, then

$$evalh(norm(x, V), N) = evalp(x, A).$$

Thus, if two polynomials produce the same normal form, then they are equivalent.

A possible top-down approach to the definition of $norm(f, V)$ is as follows:

(1) If f is an integer constant, then $norm(f,V) = f$.

(2) Suppose v_0 occurs in f. Find polynomials g and h such that $f = v_0^i \cdot g + h$, g is not divisible by v_0, and v_0 does not occur in h. If $p = norm(g,V)$ and $q = norm(h,V^{(1)})$, then

$$norm(f,V) = \texttt{(POW } i\ p\ q\texttt{)}.$$

(3) Suppose v_0 does not occur in f. Let v_i be the first variable in V that does occur in f. If $p = norm(f,V^{(i)})$, then

$$norm(f,V) = \texttt{(POP } i\ p\texttt{)}.$$

For example, consider the polynomial $4x^4y^2 + 3x^3 + 2z^4 + 5$ with variable ordering $\texttt{(}x\ y\ z\texttt{)}$. Rewriting the polynomial as

$$x^3(4xy^2 + 3) + (2z^4 + 5),$$

we find that the normalization is $\texttt{(POW 3 } p\ q\texttt{)}$, where $p = norm(4xy^2 + 3, \texttt{(}x\ y\ z\texttt{)})$ and $q = norm(2z^4 + 5, \texttt{(}y\ z\texttt{)})$. Continuing recursively, we arrive at the final result:

```
(POW 3 (POW 1 (POP 1 (POW 2 4 0)) 3)
       (POP 1 (POW 4 2 5))).
```

It may be instructive to check that the value of this SHF for the list of values $(1\ 2\ 3)$, for example, and the value of the represented polynomial for the corresponding alist, are both 207.

It is not difficult to see that a SHF generated by this procedure is indeed normal. Unfortunately, this approach is impractical because of the general difficulty of constructing the polynomials g and h in Case (2). Our preferred definition will provide a more efficient bottom-up procedure. We begin with the two basic normalizing functions *pop* and *pow*:

Definition 4.3 *Let $i \in \mathbb{N}$ and $p \in \mathscr{H}$.*

(a) If $i = 0$ or $p \in \mathbb{Z}$, then $pop(i,p) = p$.

(b) If $p = \texttt{(POP } j\ q\texttt{)}$, then $pop(i,p) = \texttt{(POP } i+j\ q\texttt{)}$.

(c) Otherwise, $pop(i,p) = \texttt{(POP } i\ p\texttt{)}$.

Definition 4.4 *Let $i \in \mathbb{N} - \{0\}$, $p \in \mathscr{H}$, and $q \in \mathscr{H}$.*

(a) If $p = 0$, then $pow(i,p,q) = pop(1,q)$.

(b) If $p = \texttt{(POW } j\ r\ 0\texttt{)}$, then $pow(i,p,q) = \texttt{(POW } i+j\ r\ q\texttt{)}$.

(c) Otherwise, $pow(i,p,q) = \texttt{(POW } i\ p\ q\texttt{)}$.

The following properties of these functions are immediate consequences of the definitions:

Lemma 4.1 *Let $i \in \mathbb{N}$, $p \in \mathscr{H}$, $q \in \mathscr{H}$, and $N \in \mathscr{L}(Z)$.*

(a) $pop(i,p) \in \mathscr{H}$ and $evalh(pop(i,p),N) = evalh(\texttt{(POP } i\ p\texttt{)},N)$.

(b) If $i \neq 0$, then $pow(i,p,q) \in \mathscr{H}$ and $evalh(pow(i,p,q),N) = evalh(\texttt{(POW } i\ p\ q\texttt{)},N)$.

We also define a ring structure on $\mathscr{H}$, Once we have computed the SHNFs for polynomials x and y, the ring operations "$\oplus$" and "$\otimes$" compute those for $\texttt{(+ } x\ y\texttt{)}$ and $\texttt{(* } x\ y\texttt{)}$.

Definition 4.5 *Let $x \in \mathscr{H}$ and $x \in \mathscr{H}$.*

(1) If $x \in \mathbb{Z}$, then

(a) $y \in \mathbb{Z} \Rightarrow x \oplus y = x + y$ *and* $x \otimes y = xy$.

(b) $y = \texttt{(POP } i\ p\texttt{)} \Rightarrow x \oplus y = \texttt{(POP } i\ x \oplus p\texttt{)}$ *and* $x \otimes y = pop(i, x \otimes p)$.

(c) $y = \texttt{(POW } i\ p\ q\texttt{)} \Rightarrow x \oplus y = \texttt{(POW } i\ p\ x \oplus q\texttt{)}$ *and* $x \otimes y = pow(i, x \otimes p, x \otimes q)$.

(2) *If* $y \in \mathbb{Z}$, *then* $x \oplus y = y \oplus x$ *and* $x \otimes y = y \otimes x$.

(3) *If* $x = \texttt{(POP } i\ p\texttt{)}$ *and* $y = \texttt{(POP } j\ q\texttt{)}$, *then*

(a) $i = j \Rightarrow x \oplus y = pop(i, p \oplus q)$ *and* $x \otimes y = pop(i, p \otimes q)$.

(b) $i > j \Rightarrow x \oplus y = pop(j, \texttt{(POP } i - j\ p\texttt{)} \oplus q)$ *and* $x \otimes y = pop(j, \texttt{(POP } i - j\ p\texttt{)} \otimes q)$.

(c) $i < j \Rightarrow x \oplus y = pop(i, \texttt{(POP } j - i\ q\texttt{)} \oplus p)$ *and* $x \otimes y = pop(i, \texttt{(POP } j - i\ q\texttt{)} \otimes p)$.

(4) *If* $x = \texttt{(POP } i\ p\texttt{)}$ *and* $y = \texttt{(POW } j\ q\ r\texttt{)}$, *then*

(a) $i = 1 \Rightarrow x \oplus y = \texttt{(POW } j\ q\ r \oplus p\texttt{)}$ *and* $x \otimes y = \texttt{(POW } j\ x \otimes q\ p \otimes r\texttt{)}$.

(b) $i > 1 \Rightarrow x \oplus y = \texttt{(POW } j\ q\ r \oplus \texttt{(POP } i - 1\ p\texttt{))}$ *and* $x \otimes y = \texttt{(POW } j\ x \otimes q\ \texttt{(POP } i - 1\ p\texttt{)} \otimes r\texttt{)}$.

(5) *If* $y = \texttt{(POP } i\ p\texttt{)}$ *and* $y = \texttt{(POW } j\ q\ r\texttt{)}$, *then* $x \oplus y = y \oplus x$ *and* $x \otimes y = y \otimes x$.

(6) *If* $x = \texttt{(POW } i\ p\ q\texttt{)}$ *and* $y = \texttt{(POW } j\ r\ s\texttt{)}$, *then*

(a) $i = j \Rightarrow x \oplus y = pow(i, p \oplus r, q \oplus s)$.

(b) $i > j \Rightarrow x \oplus y = pow(j, \texttt{(POW } i - j\ p\ 0\texttt{)} \oplus r, q \oplus s)$

(c) $i < j \Rightarrow x \oplus y = pow(i, \texttt{(POW } j - i\ q\ 0\texttt{)} \oplus p, s \oplus q)$.

(d) $x \otimes y = (pow(i + j, p \otimes r, q \otimes s) \oplus pow(i, p \otimes pop(1, s), 0)) \oplus pow(i, r \otimes pop(1, q), 0)$.

The definitions of negation and exponentiation are straightforward:

Definition 4.6 *Let* $x \in \mathscr{H}$.

(1) *If* $x \in \mathbb{Z}$, *then* $\ominus x = -x$.

(2) *If* $x = \texttt{(POP } i\ p\texttt{)}$, *then* $\ominus x = \texttt{(POP } i\ \ominus p\texttt{)}$.

(3) *If* $x = \texttt{(POW } i\ p\ q\texttt{)}$, *then* $\ominus x = \texttt{(POW } i\ \ominus p\ \ominus q\texttt{)}$.

Definition 4.7 *If* $x \in \mathscr{H}$ *and* $k \in \mathbb{N}$, *then*

$$x^k = \begin{cases} 1 & \text{if } k = 0 \\ x \otimes x^{k-1} & \text{if } k > 0. \end{cases}$$

The following properties are easily proved by induction:

Lemma 4.2 *Let* $x \in \mathscr{H}$, $y \in \mathscr{H}$, $N \in \mathscr{L}(Z)$, *and* $k \in \mathbb{N}$.

(a) $x \oplus y \in \mathscr{H}$ *and* $evalh(x \oplus y, N) = evalh(x, N) + evalh(y, N)$.

(b) $x \otimes y \in \mathscr{H}$ *and* $evalh(x \otimes y, N) = evalh(x, N) \cdot evalh(y, N)$.

(c) $\ominus x \in \mathscr{H}$ *and* $evalh(\ominus x, N) = -evalh(x, N)$.

(d) $x^k \in \mathscr{H}$ *and* $evalh(x^k, N) = evalh(x, N)^k$.

We can now define the normalization procedure:

Definition 4.8 *If* $x \in \mathscr{T}(V)$, *where* $V = \texttt{(}v_0 \ldots v_{k-1}\texttt{)}$ *is a list of distinct symbols, then*

(1) $x \in \mathbb{Z} \Rightarrow norm(x, V) = x$.

(2) $x = v_i$, $0 \leq i < k \Rightarrow norm(x, V) = pop(i, \texttt{(POW 1 1 0)})$.

(3) $x = \texttt{(- } y\texttt{)} \Rightarrow norm(x,V) = \ominus norm(y,V)$.

(4) $x = \texttt{(+ } y\ z\texttt{)} \Rightarrow norm(x,V) = norm(y,V) \oplus norm(z,V)$.

(5) $x = \texttt{(- } y\ z\texttt{)} \Rightarrow norm(x,V) = norm(y,V) \oplus (\ominus norm(z,V))$.

(6) $x = \texttt{(* } y\ z\texttt{)} \Rightarrow norm(x,V) = norm(y,V) \otimes norm(z,V)$.

(7) $x = \texttt{(EXPT } y\ k\texttt{)} \Rightarrow norm(x,V) = norm(y,V)^k$.

The reader may wish to check that the SHNF for the polynomial $-z+x^3(z+x-3y)$ with respect to the variable list $\texttt{(}x\ y\ z\texttt{)}$ is once again

```
(POW 3 (POW 1 (POP 1 (POW 2 4 0)) 3)
       (POP 1 (POW 4 2 5))).
```

Lemma 4.3 *Let $f \in \mathscr{T}(V)$, where $V = \texttt{(}v_0 \ldots v_{k-1}\texttt{)}$ is a list of distinct symbols. Let $N = \texttt{(}n_0 \ldots n_{\ell-1}\texttt{)}$ be a list of integers with $\ell \geq k$ and*

$$A = \texttt{((}v_0\ n_0\texttt{)} \ldots \texttt{(}v_{k-1}\ n_{k-1}\texttt{))},$$

Then $norm(f,V) \in \mathscr{H}$ and

$$evalh(norm(f,V),N) = evalp(f,A).$$

PROOF: The case $f = v_i$ follows from Definition 4.2 and Lemma 4.1; the other cases follow from Definition 4.2, induction, and Lemma 4.2. □

5 Polynomial Reduction

We shall focus on the case of a list of variables corresponding to the coordinates of the points P_0, P_1, and P_2, as characterized in Section 2. We define the following lists:

Definition 5.1 $\mathscr{V} = \texttt{(Y0 Y1 Y2 X0 X1 X2)}$, $\mathscr{N} = \texttt{(}y_0\ y_1\ y_2\ x_0\ x_1\ x_2\texttt{)}$, *and*

$$\mathscr{A} = \texttt{((Y0 }y_0\texttt{) (Y1 }y_1\texttt{) (Y2 }y_2\texttt{) (X0 }x_0\texttt{) (X1 }x_1\texttt{) (X2 }x_2\texttt{))}$$

We abbreviate $\mathscr{T}(\mathscr{V})$ as $\mathscr{T}$, and for $\tau \in \mathscr{T}$ we abbreviate $evalp(\tau,\mathscr{A})$ as $\hat{\tau}$.

The ordering of the variable list $\mathscr{V}$ is designed to maximize the efficiency of the rewriting procedure defined below. This procedure operates on a SHF that represents a polynomial with respect to $\mathscr{V}$, which is effectively reduced, using the curve equation as a rewrite rule, to a polynomial that (a) has the same value (modulo $\wp$) as the given polynomial under the variable assignments of $\mathscr{A}$ and (b) is at most linear in each of the variables $\texttt{Y}i$.

The core of the rewriter is the function *split*, which reduces and splits a polynomial term τ into a sum of two polynomials, of which one is independent of a given $\texttt{Y}j$ and the other is linear in $\texttt{Y}j$. More precisely, if $h = norm(\tau,\mathscr{V}^{(k)})$ and $0 \leq k \leq j \leq 3$, then $split(h,j,k)$ computes a pair of SHNFs that represent these polynomials.

The following SHNF is used in the reduction:

Definition 5.2 $\Theta = \texttt{(POP 3 (POW 1 (POW 1 (POW 1 1 }A\texttt{) 1) 0))}$.

Lemma 5.1 *If $j \in \{0,1,2\}$, then $evalh(\Theta,\mathscr{N}^{(j)}) = x_j^3 + Ax_j^2 + x_j \equiv y_j^2 \pmod{\wp}$.*

PROOF: This may be derived by expanding the definition of *evalh*. □

Definition 5.3 *Let $h \in \mathscr{H}$, $j \in \{0,1,2\}$, and $k \in \mathbb{N}$.*

(1) If $h \in \mathbb{Z}$ or $j < k$, then $split(h,j,k) = (h,0)$.

(2) If $j \geq k$, $h =$ `(POP i p)`*, and $(p_0,p_1) = split(p,j,k+i)$, then*

$$split(h,j,k) = (pop(i,p_0),pop(i,p_1)).$$

(3) Let $h =$ `(POW i p q)`*, $(p_0,p_1) = split(p,j,k)$, and $(q_0,q_1) = split(q,j,k+1)$.*

(a) If $j > k$, then

$$split(h,j,k) = (pow(i,p_0,q_0),pow(i,p_1,q_1));$$

(b) If $j = k$ and i is even, then

$$split(h,j,k) = \left((\Theta^{\frac{i}{2}} \otimes p_0) \oplus pop(1,q_0), (\Theta^{\frac{i}{2}} \otimes p_1) \oplus pop(1,q_1)\right);$$

(c) If $j = k$ and i is odd, then

$$split(h,j,k) = \left((\Theta^{\frac{i+1}{2}} \otimes p_1) \oplus pop(1,q_0), (\Theta^{\frac{i-1}{2}} \otimes p_0) \oplus pop(1,q_1)\right).$$

Lemma 5.2 *Let $(h_0,h_1) = split(h,j,k)$, where $h \in \mathscr{H}$, $j \in \{0,1,2\}$, and $k \in \mathbb{N}$. Then $evalh(h,\mathscr{N}^{(k)}) \equiv evalh(h_0,\mathscr{N}^{(k)}) + y_j \cdot evalh(h_1,\mathscr{N}^{(k)}) \pmod{\wp}$.*

PROOF: We may assume that $j \geq k$; otherwise the claim is trivial. The proof is by induction on the structure of h. The case $h =$ `(POP i p)` is straightforward:

$$\begin{aligned} evalh(h,\mathscr{N}^{(k)}) &= evalh(p,\mathscr{N}^{(k+i)}) \\ &\equiv evalh(p_0,\mathscr{N}^{(k+i)}) + y_j \cdot evalh(p_1,\mathscr{N}^{(k+i)}) \\ &= evalh(pop(i,p_0),\mathscr{N}^{(k)}) + y_j \cdot evalh(pop(i,p_1),\mathscr{N}^{(k)}) \\ &= evalh(h_0,\mathscr{N}^{(k)}) + y_j \cdot evalh(h_1,\mathscr{N}^{(k)}). \end{aligned}$$

Suppose $h =$ `(POW i p q)`. By the definition of *evalh*,

$$\begin{aligned} &evalh(h,\mathscr{N}^{(k)}) \\ &= y_k^i \cdot evalh(p,\mathscr{N}^{(k)}) + evalh(q,\mathscr{N}^{(k+1)}) \\ &\equiv y_k^i \left(evalh(p_0,\mathscr{N}^{(k)}) + y_j \cdot evalh(p_1,\mathscr{N}^{(k)})\right) + \left(evalh(q_0,\mathscr{N}^{(k+1)}) + y_j \cdot evalh(q_1,\mathscr{N}^{(k+1)})\right) \\ &= \left(y_k^i evalh(p_0,\mathscr{N}^{(k)}) + evalh(q_0,\mathscr{N}^{(k+1)})\right) + y_j \cdot \left(y_k^i evalh(p_1,\mathscr{N}^{(k)}) + evalh(q_1,\mathscr{N}^{(k+1)})\right). \end{aligned}$$

If $j > k$, then this may be written as

$$\begin{aligned} &evalh(\texttt{(POW i p}_0\ \texttt{q}_0\texttt{)},\mathscr{N}^{(k)}) + y_j \cdot evalh(\texttt{(POW i p}_1\ \texttt{q}_1\texttt{)},\mathscr{N}^{(k)}) \\ &= evalh(pow(i,p_0,q_0),\mathscr{N}^{(k)}) + y_j \cdot evalh(pow(i,p_1,q_1),\mathscr{N}^{(k)}) \\ &= evalh(h_0,\mathscr{N}^{(k)}) + y_j \cdot evalh(h_1,\mathscr{N}^{(k)}). \end{aligned}$$

We may assume, therefore, that $j = k$. If i is even, then

$$\begin{aligned} & y_k^i evalh(p_0, \mathscr{N}^{(k)}) + evalh(q_0, \mathscr{N}^{(k+1)}) \\ = \;& (y_k^2)^{\frac{i}{2}} evalh(p_0, \mathscr{N}^{(k)}) + evalh(q_0, \mathscr{N}^{(k+1)}) \\ \equiv \;& evalh(\Theta, \mathscr{N}^{(k)})^{\frac{i}{2}} evalh(p_0, \mathscr{N}^{(k)}) + evalh(\texttt{(POP 1 } q_0\texttt{)}, \mathscr{N}^{(k)}) \\ = \;& evalh(h_0, \mathscr{N}^{(k)}), \end{aligned}$$

and similarly,

$$y_j \left(y_k^i evalh(p_1, \mathscr{N}^{(k)}) + evalh(q_1, \mathscr{N}^{(k+1)}) \right) = y_j \cdot evalh(h_1, \mathscr{N}^{(k)}).$$

if i is odd, then we may rearrange the above expression for $evalh(h, \mathscr{N}^{(k)})$ as

$$\begin{aligned} & \left((y_j^2)^{\frac{i+1}{2}} evalh(p_1, \mathscr{N}^{(k)}) + evalh(q_0, \mathscr{N}^{(k+1)}) \right) \\ + \;& y_j \left((y_j^2)^{\frac{i-1}{2}} evalh(p_0, \mathscr{N}^{(k)}) + evalh(q_1, \mathscr{N}^{(k+1)}) \right), \end{aligned}$$

which similarly reduces to $evalh(h_0, \mathscr{N}^{(k)}) + y_j \cdot evalh(h_1, \mathscr{N}^{(k)})$. □

Definition 5.4 *If $h \in \mathscr{H}$, $j \in \{0,1,2\}$, and $(h_0, h_1) = split(h, j, 0)$, then*

$$rewrite(h, j) = h_0 \oplus (h_1 \otimes norm(\texttt{Y}j, \mathscr{V})).$$

Lemma 5.3 *If $h \in \mathscr{H}$, $j \in \{0,1,2\}$, and $r = rewrite(h, j)$, then $\hat{r} \equiv \hat{h} \pmod{\wp}$.*

PROOF: We instantiate Lemma 5.2 with $k = 0$ and invoke Lemma 4.2. □

Definition 5.5 *If $\sigma \in \mathscr{T}$, then*

$$reduce(\sigma) = rewrite(rewrite(rewrite(norm(\sigma, \mathscr{V}), 0), 1), 2).$$

Lemma 5.4 *If $reduce(\sigma) = reduce(\tau)$, then $\hat{\sigma} \equiv \hat{\tau} \pmod{\wp}$.*

PROOF: This is a consequence of Lemmas 5.3 and 4.3). □

6 Encoding Points on the Curve as Term Triples

The evaluation of terms induces a mapping from $\mathscr{T}^3$ to $\mathbb{F}_\wp^2$:

Definition 6.1 *For $\Pi = (\mu, \nu, \zeta) \in \mathscr{T}^3$, $\hat{\Pi} = (\hat{\mu}, \hat{\nu}, \hat{\zeta})$ and if $\hat{\zeta}$ is not divisible by $\wp$, then $decode(\Pi) = decode(\hat{\Pi})$.*

Clearly, under the following definitions, $decode(\Omega) = O$ and $decode(\Pi_i) = P_i$.

Definition 6.2 *$\Omega = (0,0,1)$ and for $i \in \{0,1,2\}$, $\Pi_i = (\texttt{X}i, \texttt{Y}i, 1)$.*

Definition 3.2 suggests a partial addition on $\mathscr{T}^3$ corresponding to the group operation on EC. This in combination with normalization (Definition 4.8) and reduction (Definition 5.5) will provide a practical means of establishing equivalence between expressions constructed from the above points by nested applications of $\oplus$, while avoiding the intractable task of explicitly computing those expressions.

Definition 6.3 *For $\Pi \in \mathscr{T}^3$ and $\Lambda \in \mathscr{T}^3$, $\Pi \oplus \Lambda \in \mathscr{T}^3$ is defined in the following cases:*

(1) *If $\Pi = \Lambda = (\mu, \nu, \zeta)$, then $\Pi \oplus \Lambda = (\mu', \nu', \zeta')$, where*

$$\zeta' = \zeta_{dbl}(\Pi) = \texttt{(* 2 (* } \nu \texttt{ } \zeta \texttt{))},$$

$$\omega = \omega_{dbl}(\Pi) = \texttt{(+ (* 3 (EXPT } \mu \texttt{ 2))}\ \texttt{(+ (* 2 (* } A \texttt{ (* } \mu \texttt{ (EXPT } \zeta \texttt{ 2))))}\ \texttt{(EXPT } \zeta \texttt{ 4))),}$$

$$\mu' = \mu_{dbl}(\Pi) = \texttt{(- (EXPT } \omega' \texttt{ 2)}\ \texttt{(* 4 (* (EXPT } \nu \texttt{ 2) (+ (* } A \texttt{ (EXPT } \zeta \texttt{ 2)) (* 2 } \mu \texttt{))))},$$

$$\nu' = \nu_{dbl}(\Pi) = \texttt{(- (* } \omega' \texttt{ (- (* 4 (* (EXPT } \nu \texttt{ 2))) } \mu' \texttt{))}\ \texttt{(* 8 (EXPT } \nu \texttt{ 4)))}.$$

(2) *If $\Pi = (\theta, \phi, 1)$ and $\Lambda = (\mu, \nu, \zeta) \neq \Pi$, then $\Pi \oplus \Lambda = (\mu', \nu', \zeta')$, where*

$$\zeta' = \zeta_{sum}(\Pi, \Lambda) = \texttt{(* } \zeta \texttt{ (- (* (EXPT } \zeta \texttt{ 2) } \theta \texttt{) } \mu \texttt{)},$$

$$\mu' = \mu_{sum}(\Pi, \Lambda) = \texttt{(- (EXPT (- (* (EXPT } \zeta \texttt{ 3) } \nu \texttt{) 2)}\ \texttt{(* (+ (* (EXPT } \zeta \texttt{ 2) (+ } A \texttt{ } \theta \texttt{)) } \mu \texttt{)}\ \texttt{(EXPT (- (* (EXPT } \zeta \texttt{ 2) } \theta \texttt{) } \mu \texttt{) 2))),}$$

$$\nu' = \nu_{sum}(\Pi, \Lambda) = \texttt{(- (* (- (* (EXPT } \zeta \texttt{ 3) } \phi \texttt{) } \nu \texttt{)}\ \texttt{(- (* (EXPT } \zeta' \texttt{ 2) } \theta \texttt{) } \mu' \texttt{))}\ \texttt{(* (EXPT } \zeta \texttt{ 3) } \phi \texttt{))}.$$

Lemma 6.1 *Let $\Pi \in \mathscr{T}^3$ and $\Lambda \in \mathscr{T}^3$ with decode$(\Pi) = P \in EC$, decode$(\Lambda) = Q \in EC$, and $\Pi \oplus \Lambda$ defined. Assume that if $\Pi = \Lambda$, then $P = Q \neq O$, and otherwise $P \neq Q$. Then decode$(\Pi \oplus \Lambda) = P \oplus Q$.*

PROOF: Let $\Gamma = \Pi \oplus \Lambda$. Clearly, the hypothesis implies that $\hat{\Pi} \oplus \hat{\Lambda}$ is defined. In light of Lemma 3.1, we need only show that $\hat{\Gamma} = \hat{\Pi} \oplus \hat{\Lambda}$.

We shall examine the case $\Pi = \Lambda$; the remaining case is similar. Let $\Pi = (\mu, \nu, \zeta)$, $\Lambda = (\mu', \nu', \zeta')$, and

$$\mathscr{P} = \hat{\Pi} \oplus \hat{\Lambda} = (\hat{\mu}, \hat{\nu}, \hat{\zeta}) = (m, n, z).$$

According to Definition 6.3,

$$\zeta' = \texttt{(* 2 (* } \nu \texttt{ } \zeta \texttt{))}$$

and it is clear from the definition of *evalp* that

$$\hat{\zeta}' = evalp(\zeta', \mathscr{A}) = 2 \cdot evalp(\nu, \mathscr{A}) \cdot evalp(\zeta, \mathscr{A}) = 2\hat{\nu}\hat{\zeta} = 2mn = z_{dbl}(\mathscr{P}).$$

It may similarly be shown that $\hat{\mu} = m_{dbl}(\mathscr{P})$ and $\hat{\nu} = n_{dbl}(\mathscr{P})$. Thus,

$$\hat{\Gamma} = (\hat{\mu}', \hat{\nu}', \hat{\zeta}') = (z_{dbl}(\mathscr{P}), m_{dbl}(\mathscr{P}), n_{dbl}(\mathscr{P})) = \mathscr{P} \oplus \mathscr{P}. \ \square$$

We also define a negation operator, with the obvious property:

Definition 6.4 *For $\Pi = (\mu, \nu, \zeta) \in \mathscr{T}^3$, $\ominus\Pi = (\mu, \texttt{(- } \nu \texttt{)}, \zeta)$.*

Lemma 6.2 *If* $\Pi \in \mathscr{T}^3$ *and* $decode(\Pi) \in EC$, *then* $decode(\ominus\Pi) = \ominus P$.

The next two lemmas, which combine the results of this section with those of Section 5, will be critical in establishing the group axioms: Lemma 6.3 for closure and Lemma 6.4 for commutativity and associativity.

Definition 6.5 *Given* $\Pi = (\mu, \nu, \zeta) \in \mathscr{T}^3$, *let*

```
τ = (- (EXPT ν 2)
       (+ (EXPT μ 3)
          (+ (* A (EXPT (* μ ζ) 2))
             (* μ (EXPT ζ 4))))).
```

Then Π *is an EC-encoding if* $reduce(\tau) = 0$.

Lemma 6.3 *If* Π *is an EC-encoding and* $P = decode(\Pi)$, *then* $P \in EC$.

PROOF: Let $\Pi = (\mu, \nu, \zeta)$, $\hat{\Pi} = (m, n, z)$, and $P = (x, y) = decode(\Pi)$. Then

$$\hat{\tau} = n^2 - (m^3 + A(mz)^2 + mz^4)$$

and

$$P = \left(\frac{\bar{m}}{\bar{z}^2}, \frac{\bar{n}}{\bar{z}^3}\right).$$

By Lemma 5.4, $\hat{\tau} \equiv 0 \pmod{\wp}$, and therefore, in the field $\mathbb{F}_\wp$,

$$\bar{n}^2 = \bar{m}^3 + A(\bar{m}\bar{z})^2 + \bar{m}\bar{z}^4.$$

Dividing this equation by $\bar{z}^6$ yields

$$y^2 = \frac{\bar{n}^2}{\bar{z}^6} = \frac{\bar{m}^3}{\bar{z}^6} + \frac{A\bar{m}^2}{\bar{z}^4} + \frac{\bar{m}}{\bar{z}^2} = x^3 + Ax^2 + x. \ \square$$

Definition 6.6 *Given* $\Pi = (\mu, \nu, \zeta) \in \mathscr{T}^3$ *and* $\Pi' = (\mu', \nu', \zeta') \in \mathscr{T}^3$, *let*

```
σ = (* μ (EXPT ζ' 2)),    σ' = (* μ' (EXPT ζ 2)),
τ = (* ν (EXPT ζ' 3)),    τ' = (* ν (EXPT ζ 3)).
```

Then $\Pi \sim \Pi' \Leftrightarrow reduce(\sigma) = reduce(\sigma')$ *and* $reduce(\tau) = reduce(\tau')$.

Lemma 6.4 *Let* $\Pi \in \mathscr{T}^3$ *and* $\Pi' \in \mathscr{T}^3$. *If* $decode(\Pi) = P \in EC$, $decode(\Pi') = P' \in EC$, *and* $\Pi \sim \Pi'$, *then* $P = P'$.

PROOF: Let $\Pi = (\mu, \nu, \zeta)$, $\Pi' = (\mu', \nu', \zeta')$, $\hat{\Pi} = (m, n, z)$, and $\hat{\Pi}' = (m', n', z')$. Then by Lemma 5.4,

$$mz'^2 = \hat{\mu}\hat{\zeta}'^2 = \hat{\sigma} \equiv \hat{\sigma}' = \hat{\mu}'\hat{\zeta}^2 = m'z^2 \pmod{\wp}$$

and

$$nz'^3 = \hat{\nu}\hat{\zeta}'^3 = \hat{\tau} \equiv \hat{\tau}' = \hat{\nu}'\hat{\zeta}^3 = n'z^3 \pmod{\wp}.$$

Thus, in the field $\mathbb{F}_\wp$,

$$P = \left(\frac{\bar{m}}{\bar{z}^2}, \frac{\bar{n}}{\bar{z}^3}\right) = \left(\frac{\bar{m}'}{\bar{z}'^2}, \frac{\bar{n}'}{\bar{z}'^3}\right) = P'. \ \square$$

7 Abelian Group Axioms

It must be shown that if $\{P,Q,R\} \subset EC$, then $P \oplus Q = Q \oplus P \in EC$ and $(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$. We may assume that the points are finite, since each of these properties is trivial otherwise, and without loss of generality, we may confine our attention to the fixed points P_0, P_1, and P_2.

Computations 1–11 below are computational results of evaluating the functions that are specified by Definitions 5.5 (term reduction), 6.3 (addition of term triples), 6.4 (negation of a term triple), 6.5 (EC-encoding recognizer), and 6.6 (equivalence of term triples). The lemmas of this section are derived from these results using the corresponding Lemmas 5.4, 6.1, 6.2, 6.3, and 6.4.

Computation 1 $\Pi_0 \oplus \Pi_0$ *and* $\Pi_0 \oplus \Pi_1$ *are EC-encodings.*

Lemma 7.1 (Closure) $P_0 \oplus P_1 \in EC$.

PROOF: If $P_0 \neq P_1$, then by Computation 1, Definition 6.2 and Lemmas 6.1,

$$P_0 \oplus P_1 = decode(\Pi_0) \oplus decode(\Pi_1) = decode(\Pi_0 \oplus \Pi_1),$$

and by Lemma 6.3, $P_0 \oplus P_1 \in EC$. Similarly, $P_0 \oplus P_0 \in EC$. □

Computation 2 $\Pi_0 \oplus \Pi_1 \sim \Pi_1 \oplus \Pi_0$.

Lemma 7.2 (Commutativity) $P_0 \oplus P_1 = P_1 \oplus P_0$.

PROOF: We may assume $P_0 \neq P_1$. By Computation 2 and Lemmas 6.1 and 6.4,

$$P_0 \oplus P_1 = decode(\Pi_0 \oplus \Pi_1) = decode(\Pi_1 \oplus \Pi_0) = P_1 \oplus P_0. \square$$

All remaining results pertain to associativity.

Computation 3 $\ominus(\Pi_0 \oplus \Pi_0) \sim (\ominus\Pi_0) \oplus (\ominus\Pi_0)$.

Computation 4 $\ominus(\Pi_0 \oplus \Pi_1) \sim (\ominus\Pi_0) \oplus (\ominus\Pi_1)$.

Lemma 7.3 $\ominus(P_0 \oplus P_1) = (\ominus P_0) \oplus (\ominus P_1)$.

PROOF: This follows from Computations 3 and 4 and Lemmas 5.4, 6.1, 6.2, and 6.4. □

Computation 5 $(\ominus\Pi_0) \oplus (\Pi_0 \oplus \Pi_0) \sim \Pi_0$.

Computation 6 $(\ominus\Pi_0) \oplus (\Pi_0 \oplus \Pi_1) \sim \Pi_1$.

Lemma 7.4 *If* $P_0 \oplus P_1 \neq \ominus P_0$, *then* $(\ominus P_0) \oplus (P_0 \oplus P_1) = P_1$.

PROOF: This follows similarly from Computations 5 and 6. □

Computation 7 $\Pi_2 \oplus (\Pi_0 \oplus \Pi_1) \sim \Pi_1 \oplus (\Pi_0 \oplus \Pi_2)$.

Computation 8 $\Pi_1 \oplus (\Pi_0 \oplus \Pi_0) \sim \Pi_0 \oplus (\Pi_0 \oplus \Pi_1)$.

Lemma 7.5 *If* $P_0 \oplus P_1 \notin \{P_2, \ominus P_2\}$ *and* $P_0 \oplus P_2 \notin \{P_1, \ominus P_1\}$, *then*

$$P_2 \oplus (P_0 \oplus P_1) = P_1 \oplus (P_0 \oplus P_2).$$

PROOF: The claim follows immediately from Computations 7 and 8 and Lemmas 5.4, 6.1, and 6.4 except in the cases $P_0 = \ominus P_1$, $P_0 = P_1 = O$, $P_0 = \ominus P_2$, and $P_0 = P_2 = O$. We need only consider the first two of these cases; the other two are similar. Moreover, since $\ominus O = O$, the second case is subsumed by the first. Thus, we may assume $P_0 = \ominus P_1$. Now *LHS* (the left-hand side) is P_2 and by Lemma 7.4, $RHS = (\ominus P_0) \oplus (P_0 \oplus P_2) = P_2 = LHS$. □

Computation 9 $(\Pi_0 \oplus \Pi_0) \oplus (\Pi_0 \oplus \Pi_0) \sim \Pi_0 \oplus (\Pi_0 \oplus (\Pi_0 \oplus \Pi_0))$.

Computation 10 $(\Pi_0 \oplus \Pi_1) \oplus (\Pi_0 \oplus \Pi_1) \sim \Pi_0 \oplus (\Pi_1 \oplus (\Pi_0 \oplus \Pi_1))$.

Lemma 7.6 *If* $P_0 \oplus P_1 \neq -(P_0 \oplus P_1)$, $P_0 \oplus P_1 \neq \ominus P_1$, *and* $P_1 \oplus (P_0 \oplus P_1) \notin \{P_0, \ominus P_0\}$, *then* $(P_0 \oplus P_1) \oplus (P_0 \oplus P_1) = P_0 \oplus (P_1 \oplus (P_0 \oplus P_1))$.

PROOF: The case $P_0 = \ominus P_1$ is trivial and the case $P_1 = P_0 \oplus P_1$ is precluded by Lemma 2.1. All other cases are handled by Computations 9 and 10 and Lemmas 5.4, 6.1, and 6.4. □

Computation 11 *Let* $\Sigma = \Pi_0 \oplus \Pi_1 = (\mu, \nu, \zeta)$, $\Sigma' = \Pi_0 \oplus \Pi_0 = (\mu', \nu', \zeta')$,

```
φ =(- (EXPT (+ (- μ (* X1 (EXPT ζ 2))) (* 2 (* Y1 Y2))) 2)
      (EXPT (* 2 (* Y1 Y2)) 2)),
```

and

```
ψ =(* (- μ' (* X2 (EXPT ζ' 2))) (EXPT ζ 2)).
```

Then $reduce(\phi) = reduce(\psi)$.

Lemma 7.7 *If* $P_0 \oplus P_1 = \ominus P_0$, *then* $P_1 = \ominus(P_0 \oplus P_0)$.

PROOF: First note that we may assume that $P_0 \notin \{P_1, \ominus P_1\}$, for if $P_0 = P_1$, then

$$P_1 = \ominus(\ominus P_0) = \ominus(P_0 \oplus P_1) = \ominus(P_0 \oplus P_0),$$

and if $P_0 = \ominus P_1$, then

$$\ominus P_0 = P_0 \oplus P_1 = (\ominus P_1) \oplus P_1 = \infty,$$

contradicting $P_0 \neq \infty$. Furthermore, if $P_0 = \ominus P_0$, then $P_0 \oplus P_1 = P_0$, contradicting Lemma 2.1. Thus, we have $x_0 \neq x_1$ and $y_0 \neq 0$.

Retaining the notation of Computation 11, let

$$\hat{\Sigma} = (\hat{\mu}, \hat{\nu}, \hat{\zeta}) = (m, n, z)$$

and

$$\hat{\Sigma}' = (\hat{\mu}', \hat{\nu}', \hat{\zeta}') = (m', n', z').$$

It follows from the definition of *evalp* that

$$\hat{\phi} = (m - x_0 z^2 + 2y_0 y_1)^2 - (2y_0 y_0)^2$$

and

$$\hat{\psi} = (m' - x_1 z'^2) z^2.$$

By Lemma 6.1,

$$decode(\Sigma) = \left(\frac{\bar{m}}{\bar{z}^2}, \frac{\bar{n}}{\bar{z}^3}\right) = P_0 \oplus P_1 = \ominus P_0 = (x_0, -y_0),$$

and hence $m \equiv x_0 z^2 \pmod{\wp}$, which implies $\hat{\phi} \equiv 0 \pmod{\wp}$.

By Computation 11 and Lemma 5.4, $\hat{\psi} \equiv 0 \pmod{\wp}$, which implies $m' \equiv x_1 z'^2 \pmod{\wp}$. Thus, by Lemma 6.1,

$$P_0 \oplus P_0 = decode(\Sigma') = \left(\frac{\bar{m'}}{\bar{z'}^2}, \frac{\bar{n'}}{\bar{z'}^3}\right) = \left(x_1, \frac{\bar{n'}}{\bar{z'}^3}\right),$$

which implies $P_0 \oplus P_0 \in \{P_1, \ominus P_1\}$. We need only consider the case $P_0 \oplus P_0 = P_1$.

Suppose that $P_0 \oplus P_0 = P_1$. Then $P_0 \oplus P_0 \neq \ominus P_0$, and by Lemma 7.10,

$$P_1 \oplus (\ominus P_0) = (P_0 \oplus P_0) \oplus (\ominus P_0) = P_0.$$

Thus,

$$P_0 \oplus (\ominus P_1) = \ominus(\ominus P_0 \oplus P_1) = \ominus(P_1 \oplus (\ominus P_0)) = \ominus P_0 = P_0 \oplus P_1.$$

By Lemma 2.2, $P_1 = O$, and hence $P_1 = \ominus P_1 = \ominus(P_0 \oplus P_0)$. □

Lemma 7.8 *If $P_0 \oplus P_1 = \ominus P_2$, then $(P_0 \oplus P_1) \oplus P_2 = P_0 \oplus (P_1 \oplus P_2)$.*

PROOF: $LHS = \infty$ and by Lemma 7.3,

$$RHS = P_0 \oplus (P_1 \oplus (\ominus(P_0 \oplus P_1))) = P_0 \oplus (P_1 \oplus ((\ominus P_0) \oplus (\ominus P_1))).$$

Therefore, we must show that $P_1 \oplus ((\ominus P_0) \oplus (\ominus P_1)) = \ominus P_0$.

If $(\ominus P_0) \oplus (\ominus P_1) \neq P_1$, then this follows from Lemma 7.4. On the other hand, if $(\ominus P_0) \oplus (\ominus P_1) = P_1$, then by Lemmas 7.7 and 7.3,

$$\ominus P_0 = \ominus((\ominus P_1) \oplus (\ominus P_1)) = P_1 \oplus P_1 = P_1 \oplus ((\ominus P_0) \oplus (\ominus P_1)).$$ □

Lemma 7.9 $(P_0 \oplus P_0) \oplus P_1 = P_0 \oplus (P_0 \oplus P_1)$.

PROOF: By Lemma 7.8, we may assume $P_0 \oplus P_1 \neq \ominus P_0$ and $P_0 \oplus P_0 \neq \ominus P_1$. By Lemmas 2.1 and 7.5, we may assume that $P_1 = P_0 \oplus P_1$.

If $P_1 = \ominus P_0$, then

$$LHS = P_1 + P_1 = (\ominus P_0) \oplus (\ominus P_0) = \ominus(P_0 \oplus P_0) = \ominus P_1 = P_0 = RHS.$$

But if $P_1 \neq \ominus P_0$, then the claim follows from Lemma 7.6. □

Two final computations are required for the case $P_1 = O$ and $P_2 = P_0 \oplus P_1$:

Computation 12 $(\Pi_0 \oplus \Omega) \oplus (\Pi_0 \oplus \Omega) \sim \Pi_0 \oplus \Pi_0$.

Computation 13 $\Omega \oplus (\Pi_0 \oplus \Omega) \sim \Pi_0$.

Lemma 7.10 $(P_0 \oplus O) \oplus (P_0 \oplus O) = P_0 \oplus (O \oplus (P_0 \oplus O))$.

PROOF: We may assume that $P_0 \neq O$. Since Lemma 2.1 implies $P_0 \oplus O \neq O$, it follows from Computation 13 that $O \oplus (P_0 \oplus O) = P_0$. Thus, the claim reduces to $(P_0 \oplus O) \oplus (P_0 \oplus O) = P_0 \oplus P_0$, which follows from Computation 12. □

Lemma 7.11 (Associativity) $(P_0 \oplus P_1) \oplus P_2 = P_0 \oplus (P_1 \oplus P_2)$.

PROOF: By Lemmas 7.6 and 7.8, we may assume $P_1 \oplus P_2 \neq \ominus P_0$, and $P_2 = P_0 \oplus P_1$. By Lemma 7.6, we need only eliminate the cases $P_0 \oplus P_1 = \ominus P_1$ and $P_1 \oplus (P_0 \oplus P_1) = P_0$.

If $P_2 = P_0 \oplus P_1 = \ominus P_1$, then $RHS = P_0$ and by Lemmas 7.3 and 7.7,

$$LHS = (P_0 \oplus P_1) \oplus (P_0 \oplus P_1) = (\ominus P_1) \oplus (\ominus P_1) = \ominus(P_1 \oplus P_1) = P_0 = RHS.$$

Finally, if $P_1 \oplus (P_0 \oplus P_1) = P_0$, then Lemma 7.9 implies $P_0 = P_0 \oplus (P_1 \oplus P_1)$, Lemma 2.1 then implies $P_1 = O$, and the claim follows from Lemma 7.10. □

References

[1] Henk Barendregt & Erik Barendsen (2002): *Autarchic Computations in Formal Proofs*. Journal of Automated Reasoning 28, pp. 321–336, doi:10.1023/A:1015761529444.

[2] Daniel J. Bernstein (2006): *Curve25519: New Diffie-Hellman Speed Records*. In: 9th International Conference on Theory and Practice of Public Key Cryptography, Springer, pp. 207–228, doi:10.1007/11745853_14.

[3] Daniel J. Bernstein & Tanja Lange (2011): *A Complete Set of Addition Laws for Incomplete Edwards Curves*. Journal of Number Theory 131, pp. 858–872, doi:10.1016/j.jnt.2010.06.015.

[4] Stefan Friedl (1998): *An Elementary Proof of the Group Law for Elliptic Curve*.

[5] Benjamin Gregoire & Assia Mahboubi (2005): *Proving Equalities in a Commutative Ring Done Right in Coq*. In: Proceedings of the 18th International Conference on Theorem Proving in Higher Order Logics, Springer-Verlag, pp. 98–113, doi:10.1007/11541868_7.

[6] Henri Poincaré (1901): *Sur les Propriétés Arithmetiques des Courbes Algébriques*. Lournalde Mathématiques Pures et Appliuées 7, pp. 121–233.

[7] Vaughn Pratt (1975): *Every Prime Has a Succinct Certification*. SIAM Journal on Computing 4, doi:10.1137/0204018.

[8] David M. Russinoff: *Polynomial Terms and Sparse Horner Normal Form*. Available at `http://www.russinoff.com/papers/shnf.pdf`.

[9] David M. Russinoff: *Pratt Certification and the Primality of* $2^{255} - 19$. Available at `http://www.russinoff.com/papers/pratt.pdf`.

[10] Joseph H. Silverman & John T. Tate (2015): *Rational Points on Elliptic Curves*. Springer-Verlag, doi:10.1007/978-3-319-18588-0.

[11] Laurent Théry (2007): *Proving the Group Law for Elliptic Curves Formally*. Technical Report RT-0330, Inria.

[12] Thomas Tymoczko (1998): *Computers and Mathematical Practice: A Case Study*. Princeton University Press.