Elliptic curve secp256k1
Implementation notes

Todor Milev*
todor@fa.biz

April 2018

1 Introdution

Public/private key cryptography is arguably the most important aspect of modern crypto-currency systems. The somewhat slow execution of private/public key cryptography algorithms appears to be one of the main bottlenecks of FAB's Kanban system.

Following Bitcoin, FAB coin uses the standard public/private key cryptography ECDSA over **secp256k1**. Here, ECDSA stands for Elliptic Curve Digital Signature Algorithm and **secp256k1** stands for the elliptic curve:

$$y^2 = x^3 + 7$$

(we specify the base point later), over the finite field:

$$\mathbb{Z}/p\mathbb{Z},$$

where

$$p = 2^{256} - 2^{32} - 977. \quad (1)$$

In this document, we discuss and document technical details of FAB's implementation of ECDSA over **secp256k1**. Our openCL implementation is based on the project [1], which is in turn based on the C project libsecp256k1 [2].

2 Operations in $\mathbb{Z}/p\mathbb{Z}$

Recall from (1) that p is the prime given by

$$p = 2^{256} - 2^{32} - 977.$$

In this section, we describe our implementation of $Z/p\mathbb{Z}$.

*FA Enterprise System

2.1 Representations of numbers

A number in x in $\mathbb{Z}/p\mathbb{Z}$ is represented by a large integer X, in turn represented by a sequence of 10 small integers $x_0, \ldots, x_9$ for which $0 \le x_i < 2^{32}$ and such that

$$X = \sum_{i=0}^{9} x_i \left(2^{26}\right)^i .$$

The representations of x is not unique but becomes so when we request that

$$0 \le x_i < 2^{26}$$

and

$$0 \le X < p = 2^{256} - 2^{32} - 977.$$

We say that the unique representation $x_0, \ldots, x_9$ of x above is its *normal form* (and x is *normalized*). Two elements of $\mathbb{Z}/p\mathbb{Z}$ are equal if and only if their normal forms are equal. We will not assume that a number x is represented by its normal form as some of the operations described below do not require that.

In what follows, we shall use the notation

$$a' = a \quad \mathbf{mod}\ q$$

to denote remainder $0 \le a' < q$ of a when dividing a by q.

2.2 Computing the normal form of x

2.3 Multiplying two elements

Let a be an element represented by the large integer A represented by $a_0, \ldots, a_9$ and b be represented by the large integer B represented by $b_0, \ldots, b_9$. In this section, we show how to compute a representation of $a \cdot b$. This operation is implemented in the function `ECMultiplyFieldElementsInner`.

Set

$$d = 2^{26}$$

and compute as follows.

$$\begin{aligned} a \cdot b &= A \cdot B \quad \mathbf{mod}\ p \\ &= \left(\sum_{i=0}^{9} a_i d^i\right) \left(\sum_{j=0}^{9} b_j d^j\right) \quad \mathbf{mod}\ p \\ &= \sum_{k=0}^{18} \left(\sum_{i=0}^{k} a_i b_{k-i}\right) d^k \quad \mathbf{mod}\ p \end{aligned}$$

Let $A \cdot B = \sum_{k=0}^{19} t_k d^k$ with $0 \le t_k < d$ be the unique representation of $A \cdot B$ base d. Set

$$\bar{t}_k = \left(\sum_{i=0}^{k} a_i b_{k-i}\right).$$

Then the t_k's can be computed from the $\bar{t}_k$'s consecutively as:

References

[1] Author: `https://github.com/hhanh00`. `https://github.com/hhanh00/secp256k1-cl` (project secp256k1-cl). 2014.

[2] Pieter Wuille and contributors. libsecp256k1 `https://github.com/sipa/secp256k1`. 2015.