UNDERSTANDING GE FROMFE FROMBYTES VARTIME

SHEN NOETHER:SHEN.NOETHER@GMX.COM- MONERO RESEARCH LABS

Abstract. I discuss the function ge fromfe frombytes vartime used in Monero's key-image functions.

Contents

1. Introduction

In this short note, I discuss the function ge-fromfe-frombytes-vartime which Monero uses for it's Key-images. Note that this code is inherited from the the original CryptoNote developers, who although seemingly competent at cryptography, have a deficiency when it comes to explaining or commenting their work. Note that I have already replaced the majority of Monero's cryptographic library last summer with Bernstein's "ref10" implementation of ed25519.

There have been several recent research papers (from well-known authors no less) discussing how to map an arbitrary string to an elliptic curve point, see [BCI+10, FFS+13]. It is interesting that the "hash to Point" function, ge-fromfe-frombytes-vartime, used in CryptoNote [vS13] does not seem to appear in any of these, and in fact seems to be potentially a more efficient algorithm.

2. FE FROMBYTES

This part is clearly fe_frombytes from ref10.

3. Unknown Part

Assume at first that $y \equiv 0$ on first try, and sign $\equiv$ sign.

Then we have:

$$2u^2 + 1 - x \equiv 0$$

so that $x \equiv 2u^2 + 1$.

Thus

$$2u^2 + 1 \equiv r_x^2 (w^2 - 2A^2u^2)$$

showing that

$$r_x = \left(\frac{2u^2 + 1}{w^2 - 2A^2u^2} \right)^{\frac{1}{2}}.$$

In this case that we have computed the square root correctly on the first try. Now we verify that the computed y and x are on the curve.

$$x_p = w^2 - 2A^2u^2 = \left(2u^2 + 1\right)^2 - 2A^2u^2$$

$$rxt = (w/x_p)^{.5}$$

$$x_t = rxt^2 \left(w^2 - 2A^2u^2\right) \to \left(\frac{w}{w^2 - 2A^2u^2} \right) \left(w^2 - 2A^2u^2\right) \to w$$

(if rxt is actuallty the square root).

$$y = \left(2u^2 + 1 - x_t\right)$$

$$rx = -u\left(2A\left(A+2\right)\frac{w}{x_p}\right)^{\frac{1}{2}} = -\left(2A\left(A+2\right)\frac{u^2w}{w^2 - 2A^2u^2}\right)^{\frac{1}{2}}$$

$$z = -2Au^2 = -\left(w-1\right)A = \left(1-w\right)A$$

(note $-z = 2Au^2$, $zA = -2A^2u^2$

$$ry = z - w$$

$$Y^2 = \left(z-w\right)^2$$

$$rz = z + w$$

$$Z^2 = \left(z+w\right)^2$$

$$r_{x-final} = \left(z+w\right)\left(2A\left(A+2\right)\frac{u^2w}{w^2+zA}\right)^{\frac{1}{2}}$$

$$\begin{aligned} X^2 &= Z^2\left(\left(A+2\right)\frac{2Au^2w}{w^2+zA}\right) \\ &= Z^2\left(A+2\right)\frac{-zw}{w^2+Az} \end{aligned}$$

$$d = -\frac{A-2}{A+2}$$

check that

$$-X^2Z^2 + Y^2Z^2 = \left(Z^2\right)^2 + dX^2Y^2$$

or in other words, that

$$Z^4\left(A+2\right)\frac{zw}{w^2+Az} + Z^2\left(z-w\right)^2 = Z^4 + \left(A-2\right)Z^2\frac{zw}{w^2+Az}\left(z-w\right)^2$$

dividing out Z^2:

$$\left(z+w\right)^2\left(A+2\right)\frac{zw}{w^2+Az} + \left(z-w\right)^2 \overset{?}{=} \left(z+w\right)^2 + \left(A-2\right)\frac{zw}{w^2+Az}\left(z-w\right)^2$$

Now multiply through by $w^2 + Az$

$$\left(z+w\right)^2\left(A+2\right)zw + \left(z-w\right)^2\left(w^2+Az\right)$$
$$\overset{?}{=} \left(z+w\right)^2\left(w^2+Az\right) + \left(A-2\right)\left(zw\right)\left(z-w\right)^2$$

After plugging in $z = \left(1-w\right)A$, you can verify with a computer algebra system, such as Maxima, that both sides are equal.

Now there are several if statements for different cases. First it is checked if the computation actually resulted in the negative square root. If this isn't the case, then it is checked if you have computed a squareroot for the negative of the original value. Finally, noting that $p = 2^255 - 19 \equiv 1 \; mod \; 4$ thus -1 is a non-residue, so taking the products of non-residues gives a residue, and we multiply our attempt by -1.

References

[BCI+10] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In *Advances in Cryptology–CRYPTO 2010*, pages 237–254. Springer, 2010.

[FFS+13] Reza R Farashahi, Pierre-Alain Fouque, Igor Shparlinski, Mehdi Tibouchi, and J Voloch. Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. *Mathematics of Computation*, 82(281):491–512, 2013.

[vS13] Nicolas van Saberhagen. Cryptonote v 2. 0. *HYPERLINK* *https://cryptonote.org/whitepaper.pdf*, 2013.