Digital Random Number Generation Using Elliptic Curves

Josephine Vo

Abstract— **For two random number generators the Power Generator, and the Naor Reingold generator, we discuss their implementation over various curves and present a comparison of the resulting performances. The curves we work with are the Weierstrass Curve and the Edwards curve. In addition we observe the effects of changing the size of the curves and mapping them to their projective representations.**

I. Introduction

Random numbers are needed in many applications of Cryptography to name a few; generation of session keys, signature keys, block ciphers, and hash functions. Elliptic curve algorithms can used to generate sequences of pseudo random numbers. These sequences are well distributed, because of the arithmetic of the elliptic curve. They are also secure, because determining information about the generator would amount to solving the ECDLP, which cannot yet be done in polynomial time. Thus using elliptic curves for RNG is very promising, especially if their computation can be done efficiently.

In this paper we begin by giving background information about the curves and the digital random number generation (DRNG) algorithms. We also detail the implementation of the DRNGs over the curves, the implementation of the curves and the method of gathering data on their performances. We use the data to compare the speed of the DRGNs over two types of curves, the Weierstrass Curve and the Edwards curves. In addition we observe the effects of changing the size of the curves and mapping them to their projective representations. We should see that for larger curves the computation over the projective curves is the faster, in particular, we should see that in with certain projections, computations over edwards curves is faster than computations over weierstrass curves.

II. Weierstrass Curves

The Weierstrass curve in this case is over a prime field, with $char(K)! = 2$, or 3, and can have the form:

$$y^2 = x^3 + ax + b$$

where a, $b \in$ K.

I implement the curve as a class with attributes that specify the curve parameters, order, and base element and also functions for adding, doubling and multiplying in the class definition. A curve can be passed into a DRNG which can access the base point, order information, and functions to generate a random number. To the top right are the algorithms for point arithmetic on the curve.

Weierstrass Add
Input: Px, Py, Qx, and Qy,
Output: Rx and Ry

1: if($P == \infty$) then return Q
2: if($Q == \infty$) then return P
3: if($Qy == Px + Py$ and $Qx == Qy$) return ∞
4: else
5: $m = Qy - Py/Qx - Px$
6: $Rx = m^2 - Px - Qx$
$Ry = m(Px - Rx) - Yp$
7: return (Rx, Ry)

Weierstrass Double
Input: (Px, Py)
Output: (Rx, Ry)

1: $m = (3(Px^2) + a)/2Py$
2: $Rx = (m^2 - 2Px)$
3: $R3 = (m(Px - R3) - Py)$
4: return $(x3, y3)$

Weierstrass Point Multiplication
Input: (Px, Py), and n
Output: (Rx, Ry)

1: let n = $n_1 n_2 ... n_m$
2: $R = \infty$
3: for i in range 0 to m :
4: if $d_i == 1$ then $(Rx, Ry) = \text{add}(Rx, Ry, x, y)$
4: else $(x, y) = \text{double}(x, y)$
5: return (Rx, Ry)

These are the point doubling and addition algorithm which are used in the point multiplication also detailed above. The point multiplication algorithm describes requires $log_2(n)$ iterations [3]. The following properties are observed:

A point addition takes **1I + 3M + 5A**
A point doubling takes **1I + 3M + 6A**

III. Edwards Curve

An Edwards curve over a prime field, with $char(K)! = 2$, has the form:

$$y^2 + x^2 = 1 + dy^2x^2$$

where $d \in$ K.

The implementation of the Edwards curve is similar to that of the Weierstrass curve except there is no need for a doubling algorithm as it is the same as the addition.

Edwards Add
Input: Px, Py, Qx, and Qy,
Output: Rx and Ry

1: if($P == (0,1)$ then return Q
2: if($Q == (0,1)$ then return P
3: if ($Px == -Qx$ and $Py == Qy$) then return $(0,0)$
4: else:
6: $Rx = (PxQy + PyQx)(1 + dPxQxPyQy)^{-1}$
7: $Ry = (PyQy - PxQx)(1 - dPxQxPyQy)^{-1}$
8: return (Rx, Ry)

Edwards Point Multiplication
Input: (Px, Py), and n
Output: (Rx, Ry)

1: let n = $n_1 n_2 ... n_m$
2: $R = \infty$
3: for i in range 0 to m :
4: if $d_i == 1$ then $(Rx, Ry) = \text{add}(Rx, Ry, x, y)$
4: else $(x, y) = \text{add}(x, y, x, y)$
5: return (Rx, Ry)

The following properties are observed:
A point addition takes **2I + 9M + 3A**

Though this is slower than the weierstrass operations, the group law is unified and this is useful for security in applications of the algorithm and is simpler to implement [2].
It is also worthwhile to note that the inversions are to most computationally intensive field operation. To work over a very large prime field we use the property that for any element a of the field Z_p, $a^{p-2} = a^{-1}$ to get the inverse. This can be made even more efficient with exponentiation by squaring (which in elliptic curves has the analogue double-and-add).

IV. Projective Coordinates

For the Weierstrass curve as previously mentioned, with the form

$$y^2 = x^3 + ax + b$$

we can define a projective form

$$Y^2 = X^3 + aXZ^4 + bZ^6.$$

The projective point (X, Y, Z) corresponds to the affine point $(X/Z^2, Y/Z^3)$, and this correspondence is 1-1. Likewise for an Edwards curve of the form

$$y^2 + x^2 = 1 + dy^2x^2$$

we can define a projective form

$$(X^2 + Y^2)Z^2 = Z^4 + dX^2Y^2.$$

A projective point (X, Y, Z) corresponds to the affine point $(X/Z, Y/Z)$ on the Edwards curve.

The projective coordinates allow for more efficient point multiplication, various formulas allow us to avoid inversions. In the end we can simply convert the point back to its affine representation if needed.

V. Power Generator

The power generator on elliptic curves is a pseudo random number function that yields a uniformly distributed sequence of random numbers [4].

We can implement the power generator on elliptic curves. Given a point on the curve, $G \in E(GF_p)$, both of which have prime order r, we randomly choose an integer e between 2 and r. Let $W_o = G$, then the sequence defined by

$$W_n = eW_{n1}, n = 1, 2, ...$$

gives us n pseudo random numbers.

VI. Naor Reingold Generator

The Naor Reingold generator is also a function on elliptic curves that produces a uniformly distributed sequence of random numbers. To define the function let E a curve of prime order r, and an element G of order l s.t. $l|r-1$. Select an n-dimensional vector $\mathbf{a} = (a_1, a_2, ..., a_n)$, where each $a_i \in Z/lZ$. Now consider the general form of the Naor Reingold an intermediate function

$$f_{a,x}(G) = a_1^{x_1} ... a_n^{x_n} G$$

where $x = x_1 x_2 .. x_n$, is the bit representation of some integer x, such that $0 \leq x \leq 2^n - 1$ with leading zeros if necessary. x and each a_i have been chosen randomly by pythons builtin DRNG. The Naor Reingold Elliptic sequence is defined as,

$$\mu_P = X(f_{a,x}(P))$$

where $X(P)$ is the perpendicular distance of a point from the vertical axis, in others words the x coordinate of a point $P \in E$.

VII. Results

The curves used to obtain the following results are: the weierstrass curves NIST P-256, NIST P-384, [7] and [6], and the edwards curves numsp256t1, numsp384t1 and numsp512t1[5].

The tables below for each algorithm show on the left columns the size of the curve and in the middle and right columns is the runtime in seconds on the DRNG. For both algorithms there is one table for its performance over affine coordinates and one table for its performance of projective coordinates.

Power Generator:

bitsize	Affine Weierstrass	Affine Edwards
256	0.0931639671326	0.339298963547
384	0.333019971848	1.09542989731
512	0.705378055573	2.80055689812

Power Generator:

bitsize	Projective Weierstrass	Projective Edwards
256	0.00321197509766	0.00147795677185
384	0.00590801239014	0.00243782997131
512	0.0104079246521	0 0.00364804267883

Naor Reingold:

bitsize	Affine Weierstrass	Affine Edwards
256	0.128608942032	3.10117793083
384	3.66616296768	6.51536607742
512	5.731372118	10.6225821972

Naor Reingold:

bitsize	Projective Weierstrass	Projective Edwards
256	0.0544121265411	0.0151789188385
384	0.0579319000244	0.00884103775024
512	0.0612258911133	0.0270869731903

The runtime for the Naor Reingold varies greatly between runs while the runtime for the power generator stays fairly consistent - within 10^{-2} figures. The cause is still unclear.

From these results it is clear to see that in the affine case the algorithms perform better over weierstrass curves than they do over edwards curves. This is consistent with our hypothesis since the weierstrass operations have one less inversion per iteration of the point multiplication. We can also see that the performance increases even more significantly through using projective coordinates by the elimination of field inversions. What is interesting here, is that in projective coordinates the edwards curves outperform the weierstrass curves. This is due to the fact that there there are no inversions to account for in either case and the projective edwards curve addition has less field multiplications than both the weierstrass curve addition and doubling.

VIII. Conclusion

There are various places in which we can make our DRNGs more efficient. In terms of the field operations, inversion is the most expensive, optimizing this helps with our performance especially when there are many inversions. However the focus of this paper was that we would like to further reduce the cost of our algorithms by eliminating inversions where it is possible.

As we have seen there are many ways to do this. The algorithms can be implemented over curves with faster group law formulas, like how the weierstrass addition and doubling is faster than the edwards addition. We can also express our curves in their projective forms to try to eliminate inversions and multiplications. As desired these optimizations resulted in faster performances of the DRNGS.

IX. Refernces

[1]Cruz, Marcos, Domingo Gmez, and Daniel Sadornil. "On the Linear Complexity of the NaorReingold Sequence with Elliptic Curves." Finite Fields and Their Applications 16.5 (2010): 329-33. Web.

[2] Enos Graham, "Binary Edwards Curves in Elliptic Curve Cryptography." (2013).

[3]Hankerson, Darrel, Alfred Menezes, and Scott Vanstone. "Guide to Elliptic Curve Cryptography." *Springer Professional Computing* (2004): n. pag. Web.

[4] Reyad Omar, and Zbigniew Kotulski. "On Pseudo-Random Number Generators Using Elliptic Curves and Chaotic Systems." *Appl. Math. Inf. Sci. Applied Mathematics amp; Information Sciences.* 9.1 (2015): 31-38. Web.

[5]C. Costello, P. Longa, and M. Naehrig. "Elliptic Curve Cryptography (ECC) Nothing Up My Sleeve (NUMS) Curves andCurve Generation." Microsoft Research. (2015): 5-7.Web.

[6] M. Lochter and J. Merkle. "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation." secunet Security Networks. (2010):10-13.Web.

[7] "RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE.".Web.