RESEARCH PAPER | Mathematics | Volume : 3 | Issue : 12 | Dec 2013 | ISSN - 2249-555X

Curve Arithmetic and Standardization in Cryptography

KEYWORDS: Elliptic curve cryptography, blum blum shub generator, finite field, montgomery's algorithm

C.Sajeev — Research Scholar , Sathyabama University

C.Suyambulingom — Professor (Rtd.), Tamilnadu Agricultural University , Coimbatore

ABSTRACT *There are many implementing procedures for cryptography to reduce the key length for providing network security. In this paper we have taken a new effort to solve this with complex analysis and used very small key for better security than ever be assigned. With four stages we conclude the arithmetic and standardization on curves. As a first phase, we generate random numbers using blum blum shub generator. After that a finite field is assigned and using with this elliptic curve has to be drawn. In which by applying montgomery's algorithm curve arithmetic and standardization is obtained.*

I INTRODUCTION

To reduce the key length and strengthen the security, we used to do it with different stages. Generation of random number is the first phase . And all the phases are as follows:

A. Generation of random number
B. Finding finite field.
C. Preparation of elliptic curves.
D. Curve arithmetic and standardization.

From the viewpoint of *x*-coordinate only arithmetic on elliptic curves, switching between the different phases is quasi cost free. We use this observation to speed up Montgomery's algorithm, reducing the complexity of a doubling step from 2**M** + 2**S** to 1**M** + 3**S** for suitably chosen curve parameters.

A. Generation of random number

Random numbers play an important role in the use of encryption for various network security applications.

There are many methodologies for generating random numbers. But in this we use Blum Blum shub Generator.

Procedures for generating random numbers

First choose two large prime numbers. Let it be p and q that both have a reminder of 3 when divided by 4.

That is

$p \equiv q \equiv 3 \pmod 4$

Prime numbers 7 and 11 satisfy 7=11=3(mod 4)
Let n=p*q =7*11=77
Next choose a random number S such that S is relatively prime to n
Consider 73 as S
That is equivalent to saying that neither p nor q is a factor of S.
Then the BBS generator produces a sequence of bits B_i
According to the following algorithm
$x_0 = S^2 \bmod n$
For i=1 to ¥
$x_i = (x_{i-1})^2 \bmod n$
$B_i = x_i \bmod 2$
Thus the last significant bit is taken at each iteration.
$x_0 = 73^2 \bmod 77$
$x_0 = 16$
$x_1 = 16^2 \bmod 77 = 25$
$B_1 = 25 \bmod 2 = 1$

i	x_i	B_i
0	16	
1	25	1

The least bits of Bi is considered up to 32-1 bit as a random number.

B. Finding finite field

A *field* is an algebraic object with two operations: addition and multiplication, represented by **+** and *****, although they will not necessarily be ordinary addition and multiplication. Using **+**, all the elements of the field must form a commutative group, with identity denoted by **0** and the inverse of **a** denoted by **-a**. Using *****, all the elements of the field except **0** must form another commutative group with identity denoted **1** and inverse of **a** denoted by $\mathbf{a^{-1}}$. (The element **0** has no inverse under *****.) Finally, the *distributive identity* must hold: **a*(b + c) = (a*b) + (a*c)**, for all field elements **a**, **b**, and**c**.

There are a number of different infinite fields, including the rational numbers (fractions), the real numbers (all decimal expansions), and the complex numbers. Cryptography focuses on *finite* fields. It turns out that for any prime integer **p** and any integer **n** greater than or equal to **1**, there is a unique field with $\mathbf{p^n}$ elements in it, denoted $\mathbf{GF(p^n)}$. (The ``GF'' stands for ``Galois Field'', named after the brilliant young French mathematician who discovered them.) Here ``unique'' means that any two fields with the same number of elements must be essentially the same, except perhaps for giving the elements of the field different names.

In case **n** is equal to **1**, the field is just the *integers mod **p***, in which addition and multiplication are just the ordinary versions followed by taking the remainder on division by **p**. The only difficult part of this field is finding the multiplicative inverse of an element, that is, given a non-zero element **a** in $\mathbf{Z_p}$, finding $\mathbf{a^{-1}}$. This is the same as finding a **b** such that **a*b % p = 1**. This calculation can be done with the extended Euclidean algorithm also.

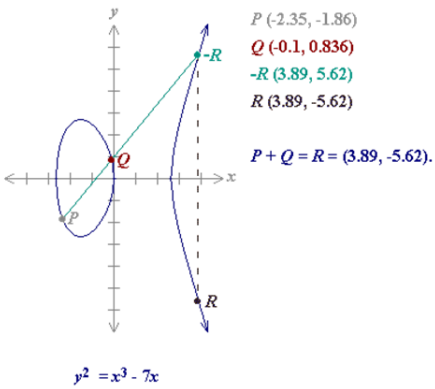
C. Preparation of elliptic curves

An elliptic curve over real numbers is a set of points (x, y) which satisfy an elliptic curve equation $y^2 = x^3 + ax + b$; where a, b, x, and y are real numbers (Certicom). The elliptic curve changes with various choices of a and b. "An elliptic curve group over real numbers consists of the points on the corresponding elliptic curve, together with a special point O

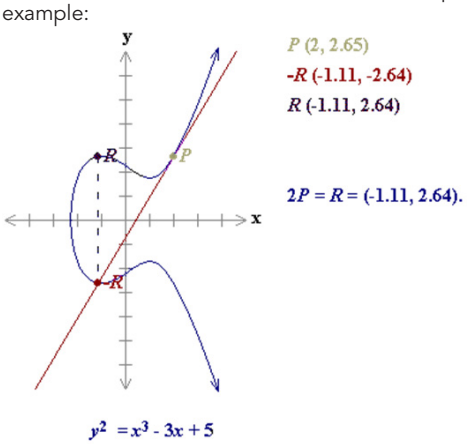
Volume : 3 | Issue : 12 | Dec 2013 | ISSN - 2249-555X

called the point at infinity" (Certicom).

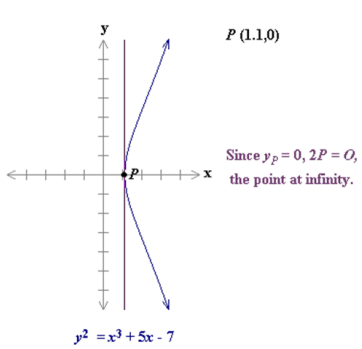
So how do we perform the addition operation on the points of an elliptic curve? You have two points, P and Q on an elliptic curve, and P + Q = R. To determine R a line is drawn through points P and Q, and the line will intersect the elliptic curve at a third point, which is –R. The point –R is then reflected in the x-axis to point R. For example:



There are two exceptions where drawing a line through points P and Q will provide point –R. The first exception occurs when adding points P and –P, the second occurs when doubling point P. Since drawing a line through point P and –P will result in a vertical line (which will not cross through the elliptic curve at a third point), the point at infinity O is needed. By definition, P + (-P) = O, therefore, P + O = P (Certicom). Now on to doubling point P. To add P to itself, a tangent line to the curve is drawn at the point P. The tangent line will intersect the elliptic curve at the point –R, if the y value of P is not 0. –R is then reflected into the x-axis to provide R. For example:



"If a point P is such that yP = 0, then the tangent line to the elliptic curve at P is vertical and does not intersect the elliptic curve at any other point. By definition, 2P = 0 for such a point P".



Now how is this done algebraically? P = (x_1, y_1) and Q = (x_2, y_2), therefore P + Q = R = (x_3, y_3). $x_3 = m^2 - x_1 - x_2$ and $y_3 = m(x_1 - x_3) - y_1$. If P ≠ Q, then $m = (y_2 - y_1) / (x_2 - x_1)$, but if P = Q, then $m = (3x_1^2 + a) / (2y_1)$. (Note: m is the slope of the line through P and Q).

Since cryptography requires that a group has a finite number of points, the finite field of integers modulo a prime number is often used. It is not possible to use the graphs of this group to "connect the dots" for the geometric relationship, but the algebraic formulas have been adapted by performing them mod p. Therefore: $x_3 = m^2 - x_1 - x_2 \pmod p$ and $y_3 = m(x_1 - x_3) - y_1 \pmod p$; and if P ≠ Q, then $m = [(y_2 - y_1) / (x_2 - x_1)] \pmod p$, but if P = Q, then $m = [(3x_1^2 + a) / (2y_1)] \pmod p$.

Given points P and Q in a group, the elliptic curve discrete logarithm problem is to find a number such that Pk = Q, for some integer k. The methods for calculating the elliptic curve discrete logarithms are less efficient than those for factoring or calculating conventional logarithms (RSA Security). Due to this, shorter key sizes can be used to provide the same level of security as other public-key cryptosystems.

D. Curve arithmetic and standardization
Montgomery's algorithm

Aiming for an improved performance of Lenstra's elliptic curve factorization method ,Montgomery developed a very efficient algorithm to compute in the group associated to an elliptic curve over a non-binary finite field F_q, in which only *x*-coordinates are involved.

The algorithm also proves useful for point compression in elliptic curve cryptography. More precisely, instead of sending a point as part of some crypto- graphic protocol, one can reduce the communication cost by sending just its *x*-coordinate. From this, the receiver can compute the *x*-coordinate of any scalar multiple using Montgomery's method. This idea was first mentioned in.

The type of curves Montgomery considered are of the following non-standard Weierstrass type

$$M_{A,B} : By^2 = x^3 + Ax^2 + x, \quad A \in F_q \setminus \{\pm 2\}, B \in F_q \setminus \{0\},$$

which is now generally referred to as a *Montgomery form*. His method works as follows. Let $P = (x1, y1, z1)$ be a point on $M_{A,B}$, the projective closure of $M_{A,B}$, and for any $n \in$ N write $n \cdot P = (xn, yn, zn)$, where the multiple is taken in the algebraic group $M_{A,B}$, ⊕ with neutral element $O = (0, 1, 0)$. Then the following recursive relations hold: for any $m, n \in$ N such that $m = n$ we have

$$xm+_n = zm-_n((xm - zm)(xn + zn) + (xm + zm)(xn - zn))^2, \text{ (ADD)}$$

$$zm+_n = xm-_n((xm - zm)(xn + zn) - (xm + zm)(xn - zn))^2.$$

and

$$4xnzn = (xn + zn)^2 - (xn - zn)^2,$$

$$x2_n = (xn + zn)^2(xn - zn)^2, \quad \text{(DOUBLE)}$$

$$z2_n = 4xnzn\,(xn - zn)^2 + ((A + 2)/4)\,(4xnzn)$$

(see also [3]). One can then compute $((xn, zn), (xn_{+1}, zn_{+1}))$ from

$$(x(n \text{ div } 2), z(n \text{ div } 2)), (x(n \text{ div } 2)+1, z(n \text{ div } 2)+1)$$

by one application of (ADD) and one application of (DOUBLE), the input of the latter depending on *n* mod 2. Thus approximately $\log_2 n$ applications of (ADD) and (DOUBLE) sufce to recover (*xn*, *zn*). Every application of (ADD) has a rough time-cost of 3**M** + **2S**, where **M** is the time needed to

Volume : 3 | Issue : 12 | Dec 2013 | ISSN - 2249-555X

multiply two general elements of F_q, and **S** is the time needed to square a general element (which is typically faster). Here we used that $z1 = 1$ in practice. Every application of (DOUBLE) needs 2**M** + **2S** + 1**C**, where **C** is the cost of multiplication of a general element of F_q with a curve constant. In this case, the constant is $(A + 2)/4$ (hence, if A is chosen carefully then **C** may be much less than **M**).

II. Conclusion

Elliptic curve cryptosystems have emerged as a promising new area in public-key cryptography in recent years," due to the smaller key sizes providing the same level of security as other public-key cryptosystems (RSA Security). As far as speed is concerned, RSA Security had this much to say: "Elliptic curve cryptosystems are faster than the corresponding discrete logarithm based systems. Elliptic curve cryptosystems are faster than the RSA system in signing and decryption, but slower in signature verification and encryption."

While the mathematical calculations on elliptical curves may be hard to understand when trying to figure out how this cryptosystem works, it is important to realize that the difficulty of the mathematical calculations is what makes this system secure. The harder the mathematical problem is to solve, the harder it should be for someone to crack the system. Unfortunately, all it takes is time and enough computers working before someone cracks it, but by then, there will hopefully be more systems that are more secure.

REFERENCE [1] Billy Bob Brumley, "Implementing Cryptography for Packet Level Authentication" billy. | [2] D. Bernstein, P. Birkner, M. Joye, T. Lange, C. Peters, Twisted Edwards Curves, AFRICACRYPT 2008, Springer Lecture Notes in Computer Science, Springer 5023, pp. 389-405 (2008) | [3] D. Bernstein and T. Lange, Faster addition and doubling on elliptic curves, Advances in Cryptology - ASIACRYPT 2007, Springer Lecture Notes in Computer Science 4833, pp. 29-50 (2007) | [4] C. Doche and T. Lange, Arithmetic of Elliptic Curves, Chapter 13 in H. Cohen and G. Frey (Eds.), Handbook of elliptic and hyperelliptic curve cryptography, Chapman & Hall/CRC Press (2005) | [5] H. Edwards, A normal form for elliptic curves. Bulletin of the American Mathematical Society 44, pp. 393-422 (2007) | [6] P. Gaudry and D. Lubicz, The arithmetic of characteristic 2 Kummer surfaces, preprint | [7] H. Lenstra, Factoring integers with elliptic curves, Annals of Mathematics 126, pp. 649-673 (1987) | [8] V. Miller, Use of elliptic curves in cryptography, CRYPTO '85, Springer Lecture Notes in Computer Science 218, pp. 417-426 (1986) | [9] P. Montgomery, Speeding the Pollard and elliptic curve methods of factorization, Mathematics of Computation 48, pp. 243-264 (1987)