CS585: Cryptography
Student: Ovidiu Mura
Email: mura@pdx.edu
Due: March 11, 2020

Elliptic Curve Cryptography (ECC)
- term paper -

Elliptic curve cryptography is used in public-key cryptography. In this paper, I will present the algebraic structure of elliptic curves over finite fields and how they are used in cryptography. The elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields.

Finite Fields

A finite field has a finite set of objects called field elements and the description of two operations, addition and multiplication. The operations can be performed on pairs of field elements and the operations must have some specific properties. Finite field F_q containing q field elements exist if and only if q is a power of a prime number and for each q there is one finite field. There are two types of finite fields: first, prime finite fields F_p with q = p an odd prime and second, characteristic 2 finite fields finite fields F_{2^m} with 2^m for some m >= 1.

Prime finite field. F_p is the prime finite field containing p elements. The elements of F_p are represented by a set of integers such as {0,1,..p-1} and there is only one F_p for each odd prime p. Operations which can be performed on F_p :

- Addition: if a,b in F_p , then a+b = r in F_p , where r in {0,..p-1} is the remainder when the integer a+b is divided by p; a+b congruent to r (mod p).
- Multiplication: if a,b in F_p , then a*b = s in F_p , where s in [0,..p-1] is the remainder when the integer a*b is divided by p; a*b congruent to s (mod p).
- Additive inverse: if a in F_p the the additive inverse (-a) of a in F_p is the unique solution to the equation a + x congruent to 0 (mod p).
- Multiplicative inverse: if a in F_p , $a \neq 0$, then the multiplicative inverse of a^{-1} of a in F_p is the unique solution to the equation a*x congruent to 1 (mod p).

Characteristic 2 finite field. F_{2^m} contains 2^m elements and there is only one F_{2^m} for each power 2^m of 2 with $m \geq 1$. F_{2^m} elements should be represented by the set of binary polynomials of degree m-1 or less, { $a_{m-1} * x^{m-1} + a_{m-2} * x^{m-2} + .. + a_1 * x + a_0 : a_i \in \{0,\ 1\}$ }.

Operations which can be performed on F_{2^m}

- Addition: if $a = a_{m-1} * x^{m-1} + ... + a_{0,} b = b_{m-1} * x^{m-1} + ... + b_0 \in F_{2^m}$, then a + b = r in F_{2^m} , where $r = r_{m-1} * x^{m-1} + ... + r_0 \, with \, r_i \equiv a_i + b_i (mod\, 2)$.
- Multiplication: if a = a_(m−1)*x^(m−1) + · · · + a_0, b = b_(m−1)*x^(m−1) + · · · + b_0 ∈ F_{2^m} , then a*b = s in F_{2^m} , where $s = s_{m-1} * x^{m-1} + ... + s_0$ is the remainder when the

polynomial ab is divided by f(x) with all coefficient arithmetic performed modulo 2.

- Additive inverse: if a ∈ F_{2^m} , then the additive inverse (−a) of a in F_{2^m} is the unique solution to the equation a + x = 0 in F_{2^m} . Note that −a = a for all a ∈ F_{2^m}.
- Multiplicative inverse: If a ∈ F_{2^m}, $a \neq 0$, then the multiplicative inverse a^{-1} of a in F_{2^m} is the unique solution to the equation ax = 1 in F_{2^m}.

For m ∈{163, 233, 239, 283, 409, 571}, we get the representation of F_{2^m}. The rule used to pick acceptable reduction polynomials was that if a degree m binary irreducible trinomial exist, $f(x)=x^m+x^k+1 \, with \, m>k\geq 1$, *k* as small as possible, otherwise we use the degree *m* binary irreducible pentanomial, $f(x)=x^m+x^{k_3}+x^{k_2}+x^{k_1}+1 \, with \, m>k_3>k_2>k_1\geq 1$, with k_3 as small as possible, k_2 as small as possible given $k_{3,}$ and k_1 as small as possible given k_3 and *k*

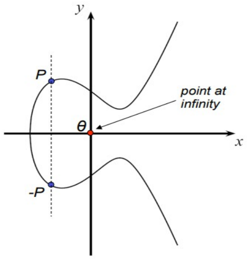
Field	Reduction Polynomial(s)
$F_{2^{163}}$	F(x) = $x^{163}+x^7+x^6+x^3+1$
$F_{2^{233}}$	F(x) = $x^{233}+x^{74}+1$
$F_{2^{239}}$	F(x) = $x^{239}+x^{36}+1 \vee x^{239}+x^{158}+1$
$F_{2^{283}}$	F(x) = $x^{283}+x^{12}+x^7+x^5+1$
$F_{2^{409}}$	F(x) = $x^{409}+x^{87}+1$
$F_{2^{571}}$	F(x) = $x^{571}+x^{10}+x^5+x^2+1$

Elliptic curve over F_p : Let F_p be a prime finite field so that p is an odd prime number, and let a, b ∈ F_p satisfy $4*a^3+27*b^2\neq 0(mod\, p)$. Then an elliptic curve $E(F_p)$ over F_p defined by the parameters a, b ∈ F_p consists of the set of solutions or points P = (x, y) for x, y ∈ F_p to the equation: $y^2\equiv x^3+a*x+b(mod\, p)$ together with an extra point O called the point at infinity. For $\# E(F_p)$ is the number of points on $E(F_p)$, the Hasse Theorem states that for a given elliptic curve module *p*, the number of points are bonded by $p+1-2\sqrt{(p)}\leq \# E\leq p+1+2\sqrt{(p)}$. The number of points is *close to* the prime *p*.

Define addition rule to add points on E:

1. Add the point at infinity to itself: O + O = O.
2. Add the point at infinity to any other point: (x, y) + O = O + (x, y) = (x, y) for all (x, y) ∈ $E(F_p)$
3. Add two points with the same x-coordinates when the points are either distinct or have y-coordinate 0: (x, y) + (x, −y) = O for all (x, y) ∈ $E(F_p);$ the negative of the point (x, y) is −(x, y) = (x, −y)

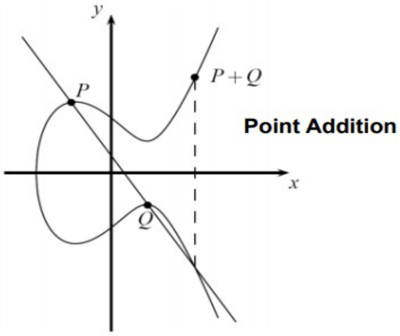
∈



∈

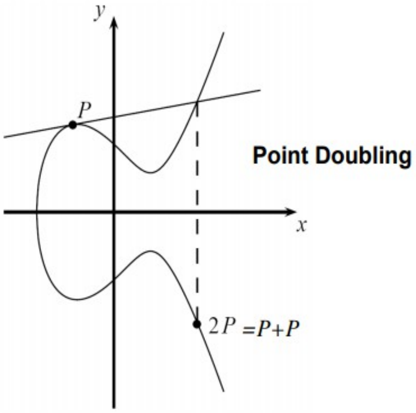
4. Add two points with different x-coordinates: Let $(x_{1,} y_1) \in E(F_p)$ and $(x_{2,} y_2) \in E(F_p)$ be two points such that $x_1 \neq x_2$. Then $(x_{1,} y_1) + (x_{2,} y_2) = (x_{3,} y_3)$, where:

   $x_3 \equiv \lambda^2 - x_1 - x_2 (mod\ p)$, $y_3 \equiv \lambda(x_1 - x_3) - y_1 (mod\ p)$, and $\lambda \equiv [(y_2 - y_1)/(x_2 - x_1)](mod\ p)$



5. Add a point to itself (double a point): Let $(x_{1,} y_1) \in E(F_p)$ be a point with $y_1 \neq 0$. Then $(x_{1,} y_1) + (x_{1,} y_1) = (x_{3,} y_3)$, where:

   $x_3 \equiv \lambda^2 - 2x^1 (mod\ p)$, $y_3 \equiv \lambda(x_1 - x_3) - y_1 (mod\ p)$, and $\lambda \equiv (3x_1^2 + a)/2y_1 (mod\ p)$



The set of points on $E(F_p)$ forms an abelian group under the addition rule, meaning that P1 + P2 = P2 + P1 for all points P1, P2 ∈ $E(F_p)$. Given an integer *k* and a point P in $E(F_p)$, scalar multiplication is the process of adding P to itself *k* times, denoted *k*P. The scalar multiplication of elliptic-curve points can be computed efficiently using the addition rule together with the double-and-add algorithm, which I will present later in this paper.

Doubling a point, P = (5,1), on a curve over the small field Z_{17}, given the elliptic-curve E: $y^2 \equiv x^3 + 2*x + 2(mod\,17)$.

$$2\text{P} = P + P = (5,1) + (5,1) = (x_{3,}y_3),$$

$$\lambda = (3*x_1^2 + a)(2*y_1) = (2*1)^{-1}*(3*5^2+2) = 2^{-1}*9 \equiv 9*9 \equiv 13\,mod\,17.$$
$$x_3 = \lambda^2 - x_1 - x_2 = 13^2 - 5 - 5 = 159 \equiv 6\,mod\,17.$$

$y_3 = \lambda(x_1 - x_3) - y_1 = 13(5-6) - 1 = -14 \equiv 3\,mod\,17.$ 2P = (5,1) + (5,1) = (6,3)

To verify that (6,3) is a point on the curve, I insert the coordinates in the curve equation:

$$y^2 \equiv x^3 + 2*x + 2\,mod\,17 \rightarrow 3^2 \equiv 6^3 + 2*6 + 2\,mod\,17 \rightarrow 9 = 230 \equiv 9\,mod\,17.$$

Discrete Logarithm Problem with Elliptic Curves

The discrete logarithm problem for elliptic curve (ECDLP) is defined as following: Let E be an elliptic curve over a finite filed F_p, where q = p^n and p is prime. Given the points P and Q ∈ $E(F_p)$ find an integer *k*, if it exists, such that Q = *k*P = P + P + .. + P , *k* times, where 1<= *k* <= #E.

This problem is the fundamental building block for elliptic curve cryptography. The discrete logarithm problem is defined over the class group of prime numbers. Another class of groups important for cryptography is given by groups consisting of *points on elliptic curves*. The groups of points on elliptic curves are very interesting for cryptography because there are not known sub-exponential time algorithms for solving the discrete-logarithm problem in elliptic-curve groups when is chosen appropriately. For example, the key length that elliptic-curve groups can be used to realize any given level of security with smaller parameters than for RSA or subgroups of multiplicative group of finite field, Z_p^*. Cryptography systems are based on the idea that *k* is large and kept secret and attackers cannot compute it easily.

An efficient algorithm for counting the number of points on an elliptic-curve is square-and-multiply algorithm, which is modified by replacing the square operation with doubling (add a point to itself) and multiplication with addition.

Double-and-Add Algorithm for point multiplication: The algorithm scans the bit representation of the scalar *k* from left to right and perform a doubling in every iteration, and only if the current bit has the value 1, it perform an addition of P.

Input: E – elliptic-curve, P – elliptic-curve point, k – a scalar $k = \sum_{i=0}^{t} k_{i2}^{i}$ with k_i ∈0,1 and $k_t = 1$, p – modulo of elliptic-curve
Output: Q = kP

```
1. Initialize Q = P
2. FOR i = t-1 DOWNTO 0
    1. Q = Q + Q mod p
    2. IF $k_i = 1$
        1. Q = Q + P mod p
3. RETURN(Q)
```

The geometric interpretation of the elliptic curve discrete logarithm is that for a given starting point P on an elliptic-curve, we compute 2P, 3P, 4P, …, *k*P = Q, moving back and forth on the elliptic-curve. The starting point P is published and the final point Q is the public-key. The cryptographic system can be broken by an attacker finding how many times we moved on the elliptic-curve. The number of hops on the elliptic-curve is the secret *k*, the private key.

There are some classes of curves considered cryptographically weak and should be avoided such as elliptic-curve groups over Z_p with order equal to p, anomalous curves, or with order equal to p+1, supersingular curves, or with order divides $p^k - 1$ for some small scalar *k*. The curves which should be used are the standardized curves recommended by NIST.

Elliptic Curve Cryptographic Schemes

Elliptic curves can be applied to key agreements, digital signatures, pseudo-random generators, integer factorization algorithms and they can be combining the key agreement with a symmetric encryption scheme to form an hybrid encryption scheme. Several discrete logarithm based protocols have been adapted to elliptic curves replacing the group Z_p^* with an elliptic curve such as the Elliptic Curve Diffie-Hellman (ECDH) key agreement scheme, the Elliptic Curve Integrated Encryption Scheme (ECIES), the Elliptic Curve Digital Signature Algorithm (ECDSA), the Edwards-curve Digital Signature Algorithm (EdDSA), the ECMQV key agreement scheme, and The ECQV implicit certificate scheme. Next, I will present the Elliptic Curve Diffie-Hellman Key Exchange.

The Elliptic Curve Diffie-Hellman Key Exchange is an anonymous key agreement scheme, which allows two parties to establish a shared secret over an insecure channel. Each party have an elliptic-curve public key and private key pair. In contrast to classical Diffe-Hellman Key Exchange (DHKE) algorithm, Elliptic Curve Diffie-Hellman (ECDH) uses elliptic curve point multiplication instead of modular exponentiation. The parties need to agree on the ECDH domain parameters. First, the both parties chose a prime p and the elliptic curve E: $y^2 \equiv x^3 + a*x + b(mod\ p)$, then they choose a primitive element $P = (x_p, y_p)$. The prime *p*, the curve with the coefficients a, b, and the primitive element P are the domain parameters.

Elliptic-Curve Diffie-Hellman is based on the following property of elliptic-curve points: (a * G) * b = (b * G) * a. Let's say Alice has the private key a secret number **a** and Bob has the private key a secret number **b** and an Elliptic Curve Cryptography with generator point G. Then the values (a * G), the public key of Alice, and (b * G), the public key of Bob, can be exchanged over an insecure channel which produce a shared secret – (a * G) * b = (b * G) * a.

The shared secret is equivalent to AlicePubKey * BobPrivKey = BobPubKey * AlicePrivKey. Next, the Elliptic Curve Diffie-Hellman Key Exchange algorithm steps are the following:

1. Alice generates a random elliptic-curve cryptography key pair (AlicePrivKey,

AlicePubKey=AlicePrivKey*G)

2. Bob generates a random elliptic-curve cryptography key pair (BobPrivKey, BobPubKey=BobPrivKey*G)
3. Alice and Bob exchange their public keys through the insecure channel
4. Alice calculates SharedKey = BobPubKey * AlicePrivKey
5. Bob calculates SharedKey = AlicePubKey * BobPrivKey
6. Alice and Bob have the same SharedKey == BobPubKey * AlicePrivKey == AlicePubKey * BobPrivKey

Parameters for elliptic curves, 160-256 bit, are significantly smaller than the RSA, 1024-3076 bit, and this result in attacks on groups of elliptic curves being weaker then available factoring algorithms or integer discrete logarithm attacks. There are known attacks against elliptic curves called Baby-Step Giant-Step and Pollard-Rho method which require roughly $\sqrt{(p)}$ steps on average to solve the *elliptic curve discrete logarithm problem* (ECDLP). For example, an elliptic-curve using a prime p with 160 bit, which can represent 2^{160} points, provides a security of 2^{80} steps that required by an attacker on average. Shor's algorithm can be used to break the elliptic-curve cryptography by computing discrete-logarithm on a hypothetical quantum computer, and ECC is easier target for quantum computers than RSA.

In this paper, I presented the prime finite fields, and the characteristic 2 finite fields and how elliptic-curve is defined over the prime finite fields. Next, I defined and presented the addition rules which can be performed on elliptic-curve. Then I discussed how discrete logarithm problem is defined for elliptic-curve, classes of curve considered cryptographically weak and an efficient algorithm to perform scalar multiplication on elliptic-curve called Double-and-Add algorithm. I presented examples of elliptic curves cryptographic schemes, and I talked in details how Elliptic Curve Diffie Hellman protocol works. Finally, I discussed the security properties of elliptic-curve in contrast to RSA, elliptic-curve provides the same security with less bits, and known successful attacks against elliptic curve cryptography.

References

[1] WikipediA free encyclopedia. (Jan 8, 2020). Elliptic-curve cryptography. WikipediA. https://en.wikipedia.org/wiki/Elliptic-curve_cryptography

[2] Brown, D. R. L. (May 21, 2009). SEC 1: Elliptic Curve Cryptography. March 9, 2020, from http://www.secg.org/sec1-v2.pdf

[3] WolframMathWorld. (Feb 17, 2020). Elliptic Curve. WolframMathWorld. March 9, 2020, from http://mathworld.wolfram.com/EllipticCurve.html

[4] Nakov, S. (November 2018). Practical Cryptography for Developers. Wizardforcel.gitbooks.io. March 9, 2020, from https://wizardforcel.gitbooks.io/practical-cryptography-for-developers-book/asymmetric-key-ciphers/elliptic-curve-cryptography-ecc.html

[5] Demos. Elliptic Curve Points. March 9, 2020, from https://www.desmos.com/calculator/ialhd71we3

[6] Paar C., Pelzl J. (2010). Understanding Cryptography. Berlin Heidelberg: Springer.

[7] Katz J., Lindell Y. (2015). Introduction To Modern Cryptography, Second Edition. Boca Raton: CRC Press Taylor & Francis Group.