Elliptic Curve Cryptography

Finite Elliptic Curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite
- have two families commonly used:
  - prime curves $E_p(a,b)$ defined over Z_p
    - use integers modulo a prime
    - best in software
  - binary curves $E_{2m}(a,b)$ defined over $GF(2^n)$
    - use polynomials with binary coefficients
    - best in hardware

Elliptic Curves over GF(2^m)

- A finite field GF(2^m) consists of 2^m elements, together with addition and multiplication that can be defined over polynomials.
- For Elliptic curves over GF(2^m), we use a cubic equation where the variables and coefficients take on the values in GF(2^m).
- The elliptic curve is of the form

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

Where x, y and a, b ε GF(2^m) and calculations are performed in GF(2^m) satisfying $4a^3 + 27b^2 \neq 0$

Elliptic Curve on a Binary field

- Consider $E2^m(a,b)$ where E: $y^2 + xy = x^3 + ax + b$

For all points P and Q on $E2^m(a,b)$

1. $P + 0 = P$
2. If $P = (xp, yp)$, then $P + (xp, xp+yp) = 0$.

   The point $(xp, xp+yp)$ is the negative of P, defined as $-P$.

3. If $P = (xp, yp)$, $Q = (xq, yq)$ with $P \neq -Q$ and $P \neq Q$, then

   $R = P + Q = (x_R, y_R)$ is determined by the following rules

   $$x_R = \lambda^2 + \lambda + xp + xq + a$$

   $$y_R = \lambda(xp + x_R) + x_R + yp$$

   where, $\lambda = (yq + yp) / (xq + xp)$

Elliptic Curve on a Binary field

- Consider $E_{2^m}(a,b)$ where E: $y^2 + xy = x^3 + ax + b$

4. If P = (xp, yp), then
   R = 2P = (x_R, y_R) is determined by the following rules

$$x_R = \lambda^2 + \lambda + a$$
$$y_R = xp + (\lambda + 1)x_R$$

where, $\lambda = xp + yp / xp$

Scalar Multiplication: MSB first

- Require $k=(k_{m-1},k_{m-2},\ldots,k_0)_2$, $k_m=1$
- Compute $Q=kP$
  - $Q=P$
  - For $i=m-2$ to 0
    - $Q=2Q$
    - If $k_i=1$ then
      - $Q=Q+P$
    - End if
  - End for
  - Return Q

- Requires m point doublings and $(m-1)/2$ point additions on the average

Example

- **Compute 7P:**
  - $7=(111)_2$
  - 7P=2(2(P)+P)+P=> 2 iterations are required
  - Principle: First double and then add (accumulate)
- **Compute 6P:**
  - $6=(110)_2$
  - 6P=2(2(P)+P)

Scalar Multiplication: LSB first

- Require $k=(k_{m-1},k_{m-2},\ldots,k_0)_2$, $k_m=1$
- Compute $Q=kP$
  - $Q=0$, $R=P$
  - For $i=0$ to $m-1$
    - If $k_i=1$ then
      - $Q=Q+R$
    - End if
    - $R=2R$
  - End for
  - Return Q

- On the average m/2 point Additions and m/2 point doublings

Example

- **Compute 7P**, $7=(111)_2$, Q=0, R=P
  - Q=Q+R=0+P=P, R=2R=2P
  - Q=P+2P=3P, R=4P
  - Q=7P, R=8P
- **Compute 6P**, $6=(110)_2$, Q=0, R=P
  - Q=0, R=2R=2P
  - Q=0+2P=2P, R=4P
  - Q=2P+4P=6P, R=8P

Weierstrass Point Addition

$$y^2 + xy = x^3 + ax^2 + b,\ (x, y) \in GF(2^m) \times GF(2^m)$$

- Let, P=(x_1,y_1) be a point on the curve.
- -P=(x_1,x_1+y_1)
- Let, R=P+Q=(x_3,y_3)

$$x_3 = \begin{cases} \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a; P \neq Q \\ x_1^2 + \frac{b}{x_1^2}; P = Q \end{cases}$$

$$y_3 = \begin{cases} \left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1; P \neq Q \\ x_1^2 + (x_1 + \frac{y_1}{x_1})x_3 + x_3; P = Q \end{cases}$$

Weierstrass Point Addition

1. Point addition and doubling each require 1 inversion & 2 multiplications

- 2. We neglect the costs of squaring and addition
- 3. ***Montgomery noticed that the x-coordinate of 2P does not depend on the y-coordinate of P***

Montgomery's method to perform scalar multiplication

- Input: $k>0$, P
- Output: $Q=kP$

1. Set $k \leftarrow (k_{l-1},\ldots,k_1,k_0)_2$
2. Set $P_1=P$, $P_2=2P$
3. For i from l-2 to 0

   If $k_i=1$,

   Set $P_1=P_1+P_2$, $P_2=2P_2$

   else

   Set $P_2=P_2+P_1$, $P_1=2P_1$
4. Return $Q=P_1$

Example

Compute 7P

- $7=(111)_2$
- Initialization:

  $P_1=P$; $P_2=2P$

- Steps:
  - $P_1=3P$, $P_2=4P$
  - **$P_1=7P$**, $P_2=8P$

Compute 6P

- $7=(110)_2$
- Initialization:

  $P_1=P$; $P_2=2P$

- Steps:
  - $P_1=3P$, $P_2=4P$
  - $P_2=7P$, **$P_1=6P$**

ECC Security

- relies on elliptic curve logarithm problem
- fastest method is "Pollard rho method"
- compared to factoring, can use much smaller key sizes than with RSA, etc.
- for equivalent key lengths computations are roughly equivalent
- hence for similar security ECC offers significant computational advantages

Applications of ECC

- Many devices are small and have limited storage and computational power
- Where can we apply ECC?
  - **Wireless communication devices**
  - Smart cards
  - Web servers that need to handle many encryption sessions
  - **Any application where security is needed but lacks the power, storage and computational power that is necessary for our current cryptosystems**

Comparable Key Sizes for Equivalent Security

Symmetric scheme (key size in bits)	ECC-based scheme (size of *n* in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360