Course Name

Title: Cyber System Security

RSA & ECC vs. SCA

F. Gebali

EOW 433

Office Phone: 250-721-6509

https://ece.engr.uvic.ca/~fayez/

Outline

Finite Fields

Finite Field

1. A set of p elements (integers) and represented by $GF(p)$ or $\mathbb{F}_p$
2. Sometimes called Galois Field
3. For any prime p and positive integer m, an extension field could be defined. The prime p is called the characteristic of the finite extension field $GF(p^m)$ or $\mathbb{F}_{p^m}$
4. Binary extension field when $p = 2$: $GF(2^m)$ or $\mathbb{F}_{2^m}$
5. Prime field when $m = 1$: $GF(p)$ or $\mathbb{F}_p$

Types of Fields

1. $\mathbb{R}$: field of real numbers
2. $\mathbb{C}$: field of complex numbers
3. $\mathbb{Z}$: field of integer numbers
4. $\mathbb{Q}$: field of rational numbers
5. $\mathbb{F}_p$: field of exactly p elements

Finite Field Properties

1. F must have the elements 0 and 1.
2. F has additive identity element 0.
3. F has multiplicative identity element 1.
4. Closure: for $a, b \in F$, $a + b \in F$ and $ab \in F$.
5. F follows distributive law:

$$a(b+c) = ab + ac$$

6. Division a/b: This implies finding q & r such that $a = qb + r$
7. Multiplicative inverse: $ab \mod p \equiv a\left(a^{-1}\right) \mod p \equiv 1$

Characteristic of F

Characteristic p of a field F is the smallest prime integer such that:

$$\underbrace{1 + 1 + ... + 1}_{p \ times} = 0$$

where 1 is the multiplicative identity and 0 is the additive identity of the field F. Of course the addition operations are all done modulo p.

Prime Field $GF(p)$ or $\mathbb{F}_p$

Prime Field $GF(p)$ or $\mathbb{F}_p$

1. $GF(p)$ is based on a prime number p and contains the integers $0, 1, 2, \cdots, p-1$.
2. Main problem with arithmetic in $GF(p)$ is carry propagation especially when p is a large number with hundreds of decimal places.

Fermat's Little Theorem for $GF(p)$

Theorem

***Fermat's Little Theorem** (FLT)* *Given $a \in GF(p)$ we can write:*

$$a^{p-1} \mod p \equiv 1 \qquad or \qquad a^p \mod p \equiv a$$

As a consequence, we have

1. $a^{p-2} \mod p = a^{-1}$
2. $a^i \mod p = \left(a^{i \mod p-1}\right) \mod p$

FLT Example

Example

Use FLT to estimate

$$x = 3^{3002} \mod 5$$

First we do modulo on exponent where mod is $p - 1 = 4$:

$$3002 \mod 4 \equiv 2$$

Therefore

$$\begin{aligned} 3^{3002} \mod 5 &= 3^{(3002 \mod 4)} \mod 5 \\ &= 3^2 \mod 5 \\ &= 9 \mod 5 = 4 \end{aligned}$$

NIST Generalized Mersenne Primes

1. $p_{192} : 2^{192} - 2^{64} - 1$
2. $p_{224} : 2^{224} - 2^{96} + 1$
3. $p_{256} : 2^{256} - 2^{224} + 2^{219} + 2^{95} - 1$
4. $p_{384} : 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
5. $p_{521} : 2^{521} - 1$

Binary Extension Field $GF(2^m)$ or $\mathbb{F}_{2^m}$

Binary Field $GF(2^m)$ or $\mathbb{F}_{2^m}$

1. $GF(2^m)$ is of order 2^m.

Binary Field $GF(2^m)$ or $\mathbb{F}_{2^m}$

1. $GF(2^m)$ is of order 2^m.
2. $GF(2^m)$ is defined with an irreducible polynmial $Q(x)$:

$$Q(x) = \sum_{i=0}^{m} q_i \, x^i$$

Binary Field $GF(2^m)$ or $\mathbb{F}_{2^m}$

1. $GF(2^m)$ is of order 2^m.
2. $GF(2^m)$ is defined with an irreducible polynmial $Q(x)$:

$$Q(x) = \sum_{i=0}^{m} q_i \, x^i$$

3. Elements in the field are polynomials having m coefficients.

Binary Field $GF(2^m)$ or $\mathbb{F}_{2^m}$

1. $GF(2^m)$ is of order 2^m.
2. $GF(2^m)$ is defined with an irreducible polynmial $Q(x)$:

$$Q(x) = \sum_{i=0}^{m} q_i\, x^i$$

3. Elements in the field are polynomials having m coefficients.
4. An element $A(x) \in GF(2^m)$ is represented as an m-term polynomial:

$$A(x) = \sum_{i=0}^{m-1} a_i\, x^i$$

where $a_i \in GF(2)$. The maximum degree of A is $m - 1$

Binary Field Irreducible (Generating) Polynomial $Q(x)$

1. $GF(2^m)$ is defined based on an irreducible polynomial $Q(x)$ of degree m

Binary Field Irreducible (Generating) Polynomial $Q(x)$

1. $GF(2^m)$ is defined based on an irreducible polynomial $Q(x)$ of degree m
2. We can write $Q(x)$ in the form:

$$Q(x) = x^m + \sum_{i=1}^{m-1} q_i\, x^i + 1$$

where $q_i \in GF(2)$.

Elements in $GF(2^3)$

An element $p(x) \in GF(2^3)$ can be written as:

$$p(x) = \alpha x^2 + \beta x + \gamma$$

$\alpha\ \beta\ \gamma$	
000	0
001	1
010	x
011	$x + 1$
100	x^2
101	$x^2 + 1$
110	$x^2 + x$
111	$x^2 + x + 1$

Fermat's Little Theorem for $GF(2^m)$

Theorem

Fermat's Little Theorem *(FLT)* *Given $p(x) \in GF(2^m)$ we can write:*

$$p(x)^{2^m} \mod Q(x) = p(x)$$

As a consequence, we have

1. $p(x)^{2^m-1} \mod Q(x) = 1$
2. $p(x)^{2^m-2} \mod Q(x) = p(x)^{-1}$
3. $p(x)^i \mod Q(x) = p(x)^{i \mod 2^m-1} \mod Q(x)$

NIST $GF(2^m)$ Irreducible Polynomials

1. $F = x^{163} + x^7 + x^6 + x^3 + 1$ (pentanomial)
2. $F = x^{233} + x^{74} + 1$ (trinomial)
3. $F = x^{283} + x^{12} + x^7 + x^5 + 1$ (pentanomial)
4. $F = x^{409} + x^{87} + 1$ (trinomial)
5. $F = x^{571} + x^{10} + x^5 + x^2 + 1$ (pentanomial)

Rivest Shamir Adelman Algorithm

Rivest-Shamir-Adleman (RSA) Cryptosystem Basic Principle

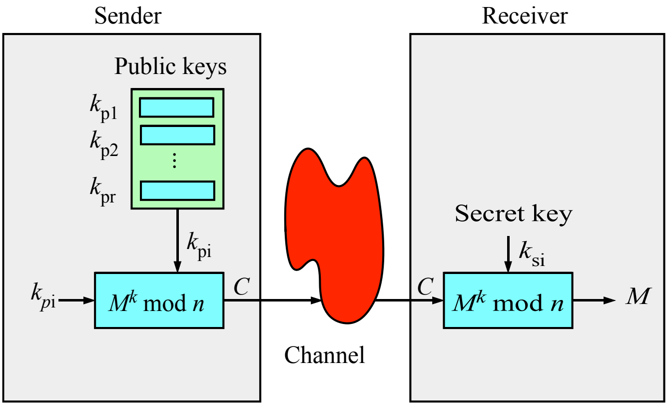
$$M^{k_s k_p} \mod n \equiv 1, \qquad C \equiv M^{k_p} \mod n, \qquad M \equiv C^{k_s} \mod n$$

1. Select different primes p, q and calculate Euler's totient:

$$n = p \times q \qquad \text{and} \qquad \lambda(n) = \text{lcm}(p-1,\ q-1)$$

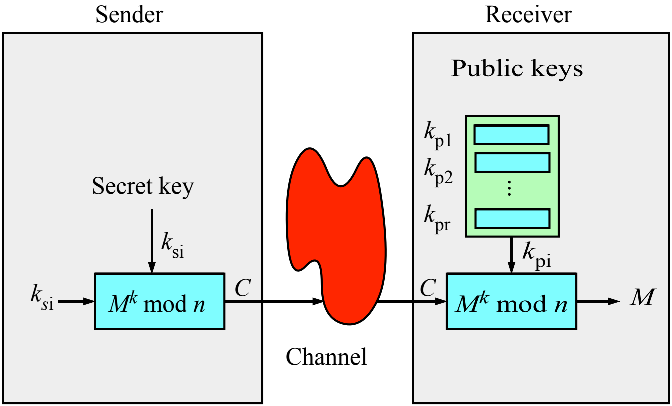
2. Select integer $2 < k_p < \lambda(n)$ such that $gcd(k_p, \lambda(n)) \equiv 1$. Usually $k_p = 2^{16} + 1$.
3. find $k_s = k_p^{-1} \mod \lambda(n)$ or $k_s\ k_p \mod \lambda(n) = 1$
4. k_s is found using extended Euclidean algorithm or Fermat little theorem (FLT)

Rivest-Shamir-Adleman (RSA): Data Privacy



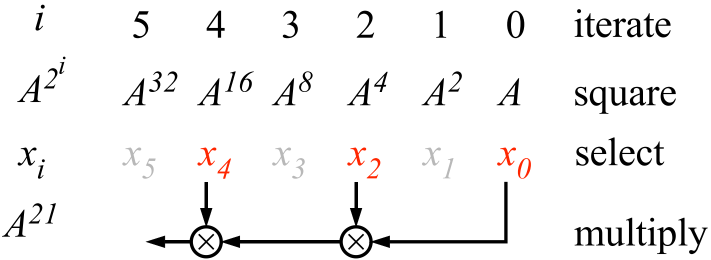
Date privacy: Only recipient can read the message

Rivest-Shamir-Adleman (RSA): Digital Signature



Digital Signature: Only sender could have generated document

Modular Exponentiation: Finding A^{21} Using Right-to-Left



Exponentiation Rules

1. $a^n = \underbrace{a \times a \times \cdots \times a}_{n \text{ terms}}$
2. $a^{b+c} = a^b \times a^c$
3. $a^{b \times c} = \left(a^b\right)^c$
4. $a^n \times b^n = (a \times b)^n$

Expressing Exponentiation in Binary

1. Assume a number A and its exponent X
2. Number of bits needed to represent X is $n = \lceil \log_2 X \rceil$
3. We can write

$$a^X = a^{x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0}$$

with $x_i \in \{0, 1\}$ for $0 \leq i < n$

Modular Exponentiation: Finding A^X Using Right-to-Left

Modular Exponentiation: Binary Representation

1. We have binary representation of exponent:

$$b = \sum_{i=0}^{k-1} b_i 2^i = \sum_{b_i \neq 0} 2^i$$

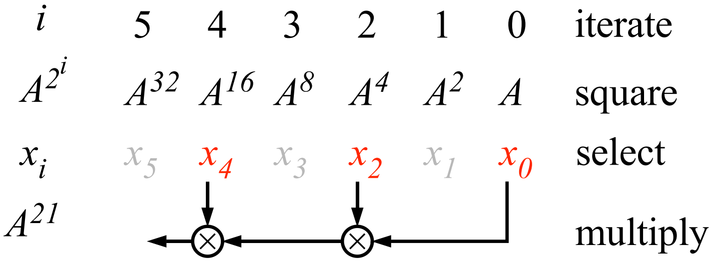
2. We have

$$A^X = \left[\prod_{x_i \neq 0} A^{2^i} \right] \mod p$$

3. We can distribute modulo inside the products:

$$A^X = \prod_{x_i \neq 0} \left[A^{2^i} \mod p \right] \mod p$$

Modular Exponentiation: Finding A^{21} Using Right-to-Left



Restrictions on $C = A^X \mod p$

We can write

$$C = A^X \mod p = \left[A^{X \mod p-1}\right] \mod p$$

We must ensure two bounds on A and X:

1. Bound on A
$$0 \leq A < p$$
2. Bound on X based on FLT
$$0 \leq X < p - 1$$

Modular Exponentiation $C = A^X \mod p$ Using Right-to-Left:
right-to-left (RtL) or LSB-to-MSB algorithm

```
Input: message A, secret key X, modulus p, and # bits B
if x(0) = 1 then
|   C = A;
else
|   C = 1;
end
for b = 1 : B − 1 do
|   A = A × A  mod p;
|       % unconditional square for next iteration
|   if x_b = 1 then
|   |   C = C × A  mod p ;
|   |           % conditional multiply
|   else
|   |   C = C;
|   end
end
```

RtL algorithm example: Case $C = A^{13} \mod 17$ Using Right-to-Left

	←	←	←	←	←
Bit index (b)	4	3	2	1	0
2^b	16	8	4	2	1
X_b	0	1	1	0	1
A	A^{16}	A^8	A^4	A^2	A
C	A^{13}	A^{13}	A^5	A	A

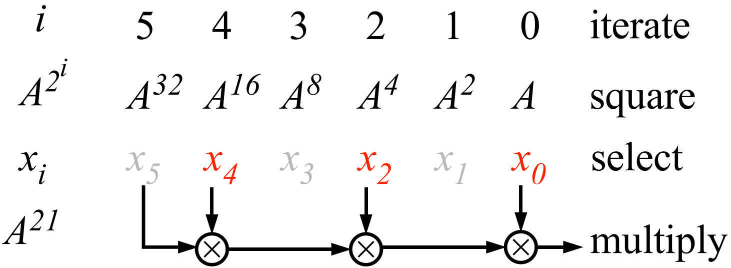
RtL algorithm example: $C = 6^{13} \mod 17$ Using Right-to-Left

	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
Bit index (b)	3	2	1	0
2^b	8	4	2	1
x	1	1	0	1
A	16	4	2	6
C	10	7	6	6

$C = 6^{13} \mod 17 = 10$

Modular Exponentiation: Finding A^X Using Left-to-Right

Modular Exponentiation: Finding A^{21} Using Left-to-Right



Modular Exponentiation $C = A^X \mod n$: left-to-right (LtR) algorithm or MSB-to-LSB

```
if x_{B-1} = 1 then
    C = A
else
    C = 1
end if
for b = B - 2 : 0 do
    C = C × C mod p
    if x_b = 1 then
        C = C × A mod p
    end if
end for
return C
```

Unconditional square C then conditional multiply by A

LtR algorithm example: Case X = 29

	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
Bit index (b)	4	3	2	1	0
2^b	16	8	4	2	1
X_b	1	1	1	0	1
C	A	A^3	A^7	A^{14}	A^{29}

LtR algorithm example: $C = 5^{27} \mod 7$

	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
Bit index (b)	4	3	2	1	0
2^b	16	8	4	2	1
X_b	1	1	0	1	1
C	5	6	1	5	6

Non-Adjacent Form (NAF)

NAF

Hopefully speeds up algorithm.

6-bit Binary number		0	1	1	1	0	1
7-bit NAF Representation	0	1	0	0	-1	0	1

Must add one extra bit on left: $B \rightarrow B + 1$

Right-to-Left NAF for Modular Exponentiation

RtL NAF Modular Exponentiation $C = A^X \mod p$: right-to-left (RtL)

$C = 1$, $A_1 = A$, $A_2 = A^{-1}$
for $i = 0 : B - 1$ **do**
    $A_1 = A_1^2 \mod p, \qquad A_2 = A_2^2 \mod p$
    if $x_i = 1$ **then**
        $C = C \times A_1 \mod p$
    else if $x_i = -1$ **then**
        $C = C \times A_2 \mod p$
    end if
end for
return C

Need to find multiplicative inverse A_2

NAF Modular Exponentiation $C = A^X \mod n$: right-to-left Example when $X = 13$

	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
X_{NAF}	1	0	-1	0	1
A_1	A^{16}	A^8	A^4	A^2	A
A_2	A^{-16}	A^{-8}	A^{-4}	A^{-2}	A^{-1}
C	A^{13}	A^{-3}	A^{-3}	A	A

NAF RtL: Case $C = 8^{27} \mod 31$

	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
Bit index (b)	5	4	3	2	1	0
2^b	32	16	8	4	2	1
X_{bin}	0	1	1	0	1	1
X_{NAF}	1	0	0	-1	0	-1
A_1	2	8	16	4	2	8
A_2	16	4	2	8	16	4
C	2	1	1	1	4	4

Left-to-Right NAF for Modular Exponentiation

LtR NAF Modular Exponentiation $C = A^X \mod n$: Left-to-Right

Require: $A, n \in \mathbb{Z}^+$, $X = (x_{B-1}, \cdots, x_1, x_0)_{NAF}$

$A_1 = A$, $A_2 = A^{-1}$, $C = 1$
if $x_{B-1} = 1$ **then**
    $C = A_1$
else if $x_{B-1} = -1$ **then**
    $C = A_2$
end if
for $i = n - 2 : 0$ **do**
    $C = C^2 \mod p$
    if $x_i = 1$ **then**
        $C = C \times A_1 \mod p$
    else if $x_i = -1$ **then**
        $C = C \times A_2 \mod p$
    end if
end for

NAF Modular Exponentiation: Left-to-Right Example $C = A^{25}$

Note we need here to increase the number of bits from 5 to 6:

	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
b	5	4	3	2	1	0
2^b	32	16	8	4	2	1
x_{NAF}	1	0	-1	0	0	1
C	A	A^2	A^3	A^6	A^{12}	A^{25}

NAF RtL Modular Exponentiation: Case $C = 12^6 \mod 17$

We have $A_1 = 12$ and $A_2 = 10$.

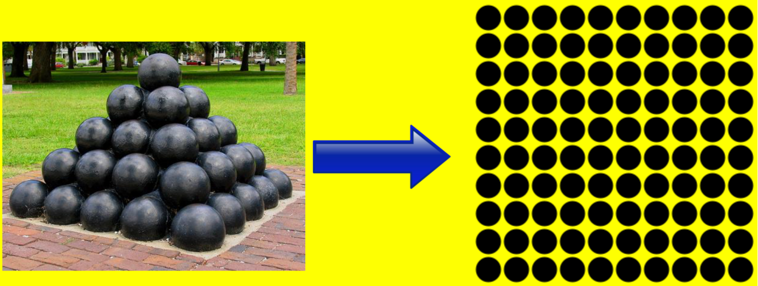
	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
Bit index (b)	3	2	1	0
2^b	8	4	2	1
x_{NAF}	1	0	-1	0
C	12	8	11	2

Elliptic Curve Cryptography

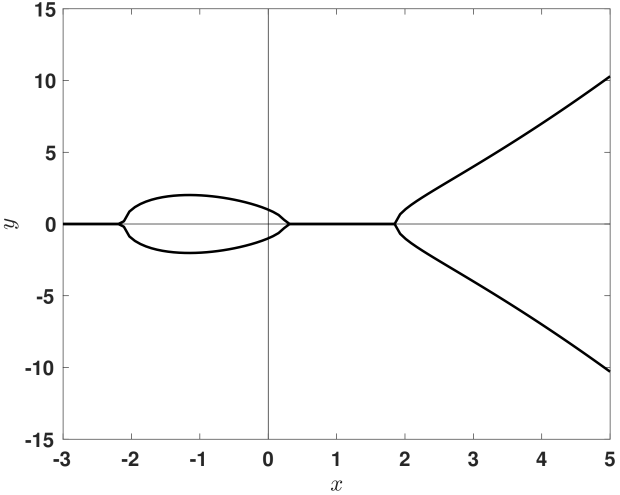
Motivation

1. ECC, as PKI, offers higher level of security for same number of secret key bits compared to RSA
2. Energy to break RSA enough to boil spoonful of water. Energy to break ECC needs to boil all water on earth.
3. ECC is suited to embedded and IoT devices with limited resources
4. Pairing ECC offers immunity to quantum attacks

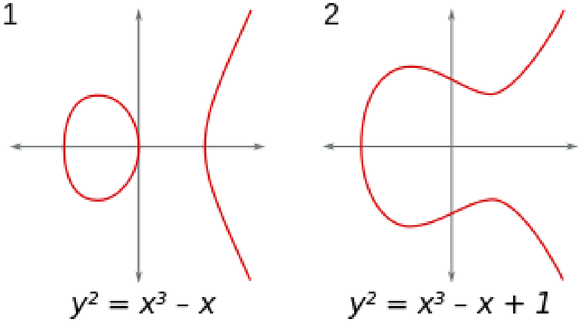
Elliptic Curve Equation Physical Origins



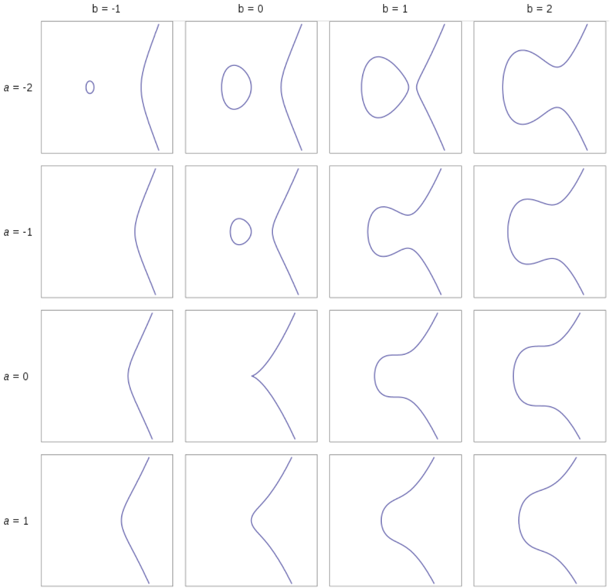
ECC Graphic Representation



Elliptic Curves



Elliptic Curves



Elliptic Curves

1. Elliptic curves of interest are a group of points over $GF(p)$ or $GF(2^m)$
2. An extra point is $\mathcal{O}$ the point at infinity so that all vertical lines pass through it
3. A point on the elliptic curve is $P = (x, y)$
4. Inverse of a point: $-P = (x, -y)$
5. $y^2 = Poly(x)$ where $Poly(x)$ is a polynomial of degree 3 with no repeated roots

Elliptic Curve Discrete Logarithm Problem

1. Given Points $P,\ Q \in E_p(a, b)$ and integer k
2. Do scalar multiplication $Q = k \times P$:

$$Q = \underbrace{P + P + \cdots + P}_{k-terms}$$

3. Publish P and Q
4. Very difficult to figure out the secret value k

ECC Discrete Logarithm Problem

1. Given $E_p(a, b)$ over $GF(p)$

$$P_2 = kP_1$$

2. It is difficult to find integer k given P_1 and P_2

ECC Cryptography

Basic ECC Cryptography Steps

1. Secret key generation
2. Data Encryption
3. Data Decryption

ECC: Secret Key Generation

1. Pick a curve $E_p(a, b)$
2. Choose a random number as secret key $0 < k_s < p$
3. Choose a generator point P_1
4. Do scalar multiplication

$$P_2 = k_s P_1$$

5. Scalar multiplication gives the public key

$$k_p = P_2 = k_s P_1$$

ECC: Encryption

1. Choose message to send M, a string of bits
2. Map M to a point m on the curve $E_p(a, b)$
3. Generate a random number d to get

$$C_1 = dP_1 \qquad \text{and} \qquad C_2 = m + dP_2$$

4. Send the two points C_1 and C_2
5. Publish public key $k_p = P_2$

ECC: Decryption

1. We get the point m as

$$m = C_2 - k_s C_1$$

2. We can write

$$\begin{aligned} m &= m + dP_2 - k_s dP_1 \\ &= m + dk_s P_1 - k_s dP_1 \end{aligned}$$

3. Thus we recovered m

4. Do inverse mapping to get M

Key Length for ECC Security

	Security Level (bits)				
	80	112	128	192	256
	SKIPJACK	3DES	AES-S	AES-M	AES-L
EC $GF(p)$	192	224	256	384	521
EC $GF(2^m)$	163	233	283	409	571
RSA	1,024	2,048	3,072	8,192	15,360

ECC Defining Equation for $GF(p)$

1. Weierstrass equation [1, 2]:

$$y^2 \mod p = x^3 + ax + b \mod p$$

where x, y, a and b are elements in $GF(p)$

2. A more concise way of writing the above equation is:

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

3. Condition for curve to be smooth:

$$\Delta = 4a^3 + 27b^2 \neq 0 \mod p$$

4. Hesse-Weil formula (number of points in E)

$$p + 1 - 2\sqrt{p} \leq \#(E) \leq p + 1 + 2\sqrt{p}$$

NIST Standard Elliptic Curve Primes for $GF(p)$

$$
\begin{aligned}
p_{192} &= 2^{192} - 2^{64} - 1 \\
p_{224} &= 2^{224} - 2^{96} + 1 \\
p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
p_{384} &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \\
p_{521} &= 2^{521} - 1
\end{aligned}
$$

Quasi Mersenne Integers

Non NIST Curves [3]: M (montgomery)

$$y^2 = x^3 + ax^2 + x \mod p \qquad \text{plus} \qquad O$$

$$\begin{aligned} p_{221} &= 2^{221} - 3, & a = 117,050 \\ p_{255} &= 2^{255} - 19, & a = 486,662 \\ p_{383} &= 2^{383} - 187, & a = 2,065,150 \\ p_{511} &= 2^{511} - 187, & a = 530,438 \end{aligned}$$

Example

max: 1157920892103562487626974469494075735300861
43415290 3141955336313088670978539 51

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

a = 1157920892103562487626974469494075735300861 4
34152903 1419553363130886709785394 8

b = 4105836372515214212932612978004726840911444 1
015993 7255548352563140394674012 91

ECC Defining Equation for $GF(2^m)$

1. Weierstrass equation [1, 2]:

$$y^2 + xy = x^3 + ax^2 + b \qquad \text{plus} \qquad \mathcal{O}$$

   where x, y, a and b are elements in $GF(2^m)$. For cryptographic purposes the value of m takes values $m > 160$.

2. Binary extension field $GF(2^m)$ does not require carry

3. Addition is same as subtraction

Example of $E \in GF(2^4)$

$$y^2 + xy = x^3 + ax^2 + 1 \qquad a \in GF(2^m) \qquad \text{plus} \qquad \mathcal{O}$$

NIST Standard Elliptic Curve Polynomials for $GF(2^m)$

$$
\begin{aligned}
f_{163}(x) &= x^{163} + x^7 + x^3 + 1 \\
f_{233}(x) &= x^{233} + x^{74} + 1 \\
f_{283}(x) &= x^{283} + x^{12} + x^7 + x^5 + 1 \\
f_{409}(x) &= x^{409} + x^{87} + 1 \\
f_{571}(x) &= x^{571} + x^{10} + x^5 + x^2 + 1
\end{aligned}
$$

Elliptic Curve Example E over $GF(23)$

Elliptic curve E over $GF(23)$ satisfying the equation:

$$y^2 = x^3 + x \quad (1)$$

1. We have $a = 1$ and $b = 0$. It can be easily verified that the point (0,0) lies on the curve.
2. The point (9,5) also lies on the curve. The left hand side is:

$$y^2 = 25 \mod 23 \equiv 2$$

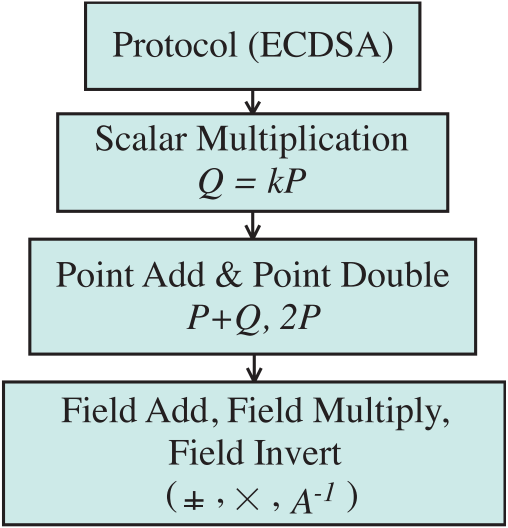
3. The right hand side produces:

$$x^3 + x = 729 + 9 \equiv 2$$

NIST Standard Elliptic Curve Polynomials for $GF(2^m)$

$$
\begin{aligned}
f_{163}(x) &= x^{163} + x^7 + x^3 + 1 \\
f_{233}(x) &= x^{233} + x^{74} + 1 \\
f_{283}(x) &= x^{283} + x^{12} + x^7 + x^5 + 1 \\
f_{409}(x) &= x^{409} + x^{87} + 1 \\
f_{571}(x) &= x^{571} + x^{10} + x^5 + x^2 + 1
\end{aligned}
$$

ECC Hierarchy



Elliptic Curve Cryptography

Scalar Multiplication

Elliptic Curve Cryptography: Scalar Multiplication

$$Q = kP, \qquad Q, P \in E, \qquad k \in \mathbb{Z}$$

1. This is the discrete logarithm problem
2. System relies on security of elliptic curve encryption
3. Given P and Q, it is very difficult to find k

ECC Scalar Multiplication $Q = kP$: Point Double & Add

ECC point doubling at iteration i:

$$kP \rightarrow \quad 2^i P = 2^{i-1} P + 2^{i-1} P$$
$$= 2 \times 2^{i-1} P$$

i.e. simple point doubling of previous result

ECC Scalar Multiplication: $Q = kP$: right-to-left (RtL) or LSB to MSB algorithm

Require: P and $k = (k_{m-1}, K_{m-2}, \cdots, k_0)$

$A = P$; $Q = \mathcal{O}$
for $i = 0 : m - 1$ **do**
    if $k_i = 1$ **then**
        $Q = Q + A$ (Conditional point add)
    end if
    $A = 2A$ (Unconditional point double for next iteration)
end for
RETURN Q

Point Doubling RtL Example: Case k = 13

	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
Bit weight (i)	4	3	2	1	0
2^i	16	8	4	2	1
k_i	0	1	1	0	1
A	$16P$	$8P$	$4P$	$2P$	P
Q	$13P$	$13P$	$5P$	P	P

Scalar Multiplication $Q = kP$: left-to-right (LtR) algorithm or MSB to LSB

Require: P and $k = (k_{m-1}, K_{m-2}, \cdots, k_0)$

$Q = P$
for $i = m-2 : 0$ **do**
  $Q = 2Q$ (Point double)
  if $k_i = 1$ **then**
    $Q = Q + P$ (Point add & double)
  else
    $Q = Q$ (Point double)
  end if
end for

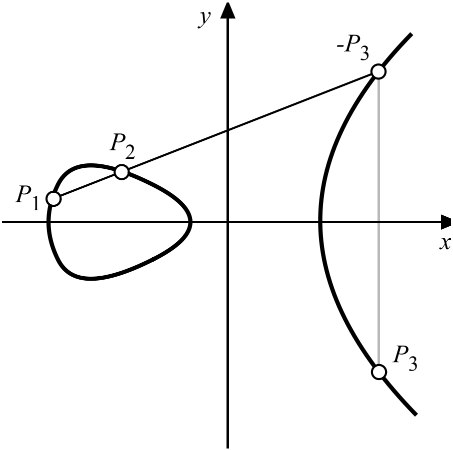
Unconditional double Q then conditional add P

LtR algorithm example: Case k = 29

	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
Bit weight (i)	4	3	2	1	0
2^i	16	8	4	2	1
k_i	1	1	1	0	1
Q	P	$3P$	$7P$	$14P$	$29P$

ECC Add

ECC Add Operation: $P_3 = P_1 + P_2$



The addition operation is written as:

$$P_3 = P_1 + P_2$$

ECC Add Operation: $P_3 = P_1 + P_2$

1. Assume $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_3 = (x_3, y_3)$
2. $$x_3 = m^2 - x_1 - x_2 \mod p$$ $$y_3 = m(x_1 - x_3) - y_1 \mod p$$

   where

   $$m = \frac{y_2 - y_1}{x_2 - x_1} \mod p$$

   m is the slope of the line connecting points P and Q.

ECC Add Operation: $P_3 = P_1 + P_2$

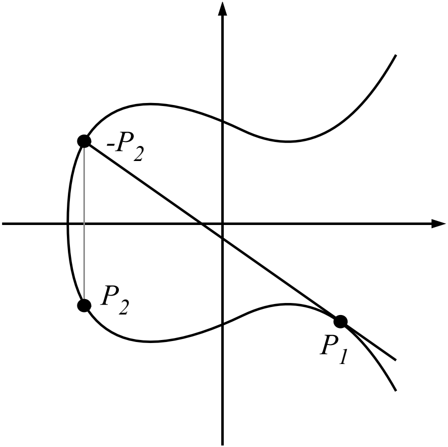
Point addition in ECC arithmetic requires the following basic finite field operations:

1. Field additions/subtractions
2. Field multiplications
3. Finding inverse of an element
4. Modular exponentiation

It is up to the system designer to implement these operations in hardware or software depending on the capabilities of the field arithmetic unit (FAU) being designed.

ECC Double

ECC Double



ECC Point Doubling Operation: $P_3 = 2P_1$

1. Assume $P_1 = (x_1, y_1)$ and $P_3 = (x_3, y_3)$,
2. $$m = \frac{3x_1^2 + a}{2y_1} \mod p$$
3. $$\begin{aligned} x_3 &= m^2 - 2x_1 \mod p \\ y_3 &= m(x_1 - x_3) - y_1 \mod p \end{aligned}$$

Elliptic Curve Calculators

1. To do point addition and multiplication in $\mathbb{R}$ and $\mathbb{F}_p$
   https://andrea.corbellini.name/ecc/interactive/reals-add.html

2. Elliptic Curves over $\mathbb{F}_|$ showing point additions
   https://graui.de/code/elliptic2/

3. Plot elliptic curve points in $\mathbb{F}_p$
   https://asecuritysite.com/encryption/ecc_pointsv?a0=0&a1=7&a2=802283

[1] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. New York: Springer, 2004.

[2] IEEE P1363.2 working group, “Standard for identity-based public-key cryptography using pairings,” *IEEE*, 2010.

[3] D. F. Aranha, P. S. L. M. Barreto, G. C. C. F. Pereira, and J. E. Ricardini, “A note on high-security general-purpose elliptic curves,” Cryptology ePrint Archive, Report 2013/647, 2013, https://eprint.iacr.org/2013/647.