CINVESTAV

ECC (Elliptic Curve Cryptography)

Francisco Rodríguez Henríquez

Introduction

CINVESTAV

- As applied to cryptography, elliptic curves were first proposed in 1985 independently by N. Koblitz, and V. Miller.
- Elliptic curves as algebraic/geometric entities have been around since the second half of the *XIX* century. Originally investigated for purely aesthetic reasons, now they are utilized in devising algorithms for factoring integers, primality tests, and in public-key cryptography.
- Elliptic curves can be defined over any field, like real numbers, complex numbers, etc. However, for cryptographic purposes we are only concern with those over finite fields, More specifically, for the rest of this talk, we will only consider binary elliptic curves, i.e., elliptic curves over $GF(2^m)$

DSA Vs. ECDSA

CINVESTAV

Group	Z	$E(Z_p)$
Group elements	Integers $\{ 1, 2, \ldots, p - 1 \}$	Points (x, y) on E plus O
Group operation	multiplication modulo p	addition of points
Notation	Elements: g, h Multiplication: $g \bullet h$ Inverse: g^{-1} Division: g / h Exponentiation: g^a	Elements: P, Q Addition: $P + Q$ Negative: $-P$ Subtraction: $P - Q$ Multiple: aP
Discrete Logarithm Problem	Given $g \in Z$ and and $h = g^a \bmod p$, find a	Given $P \in E(Z_p)$ and $Q = aP$, find a.

DSA Vs. ECDSA

DSA notation	ECDSA notation
q	n
g	P
x	d
y	Q

ECC and ECDSA: Coincidences

CINVESTAV

1. Both algorithms are based on the ElGamal signature scheme and use the same signing equation: $s = k\text{-}1\{h(m) + dr\} \bmod n$.
2. In both algorithms, the values that are difficult to generate are the system parameters (p, q and g for the DSA; p, E, P and n for the ECDSA) which are public; their generation can be audited and independently checked for validity. This helps show that they were not produced to meet some secret (e.g., trapdoor) criteria.

ECC and ECDSA: Coincidences

Generating a private key, given a set of system parameters, is relatively simple and generating the associated public key is straightforward. Contrast this with the RSA algorithm, where the values that are difficult to generate (the primes *p* and *q)* must be kept private.

3. In their current version, both DSA and ECDSA use the SHA-1 as the sole cryptographic hash function. This may be modified in the future by, for example, allowing a hash function which offers output values of variable lengths.

Elliptic curve cryptosystems

CINVESTAV

Applications: e-commerce, smart cards, digital money, secure communications, etc.

Elliptic curve protocols: Diffie-Hellman, authentication protocols, etc.

Elliptic curve primitives: Key-pair generation, Signing and Verification

Elliptic Curve operations: addition, doubling, scalar multiplication

Finite field operations : Addition, Squaring, multiplication and inversion

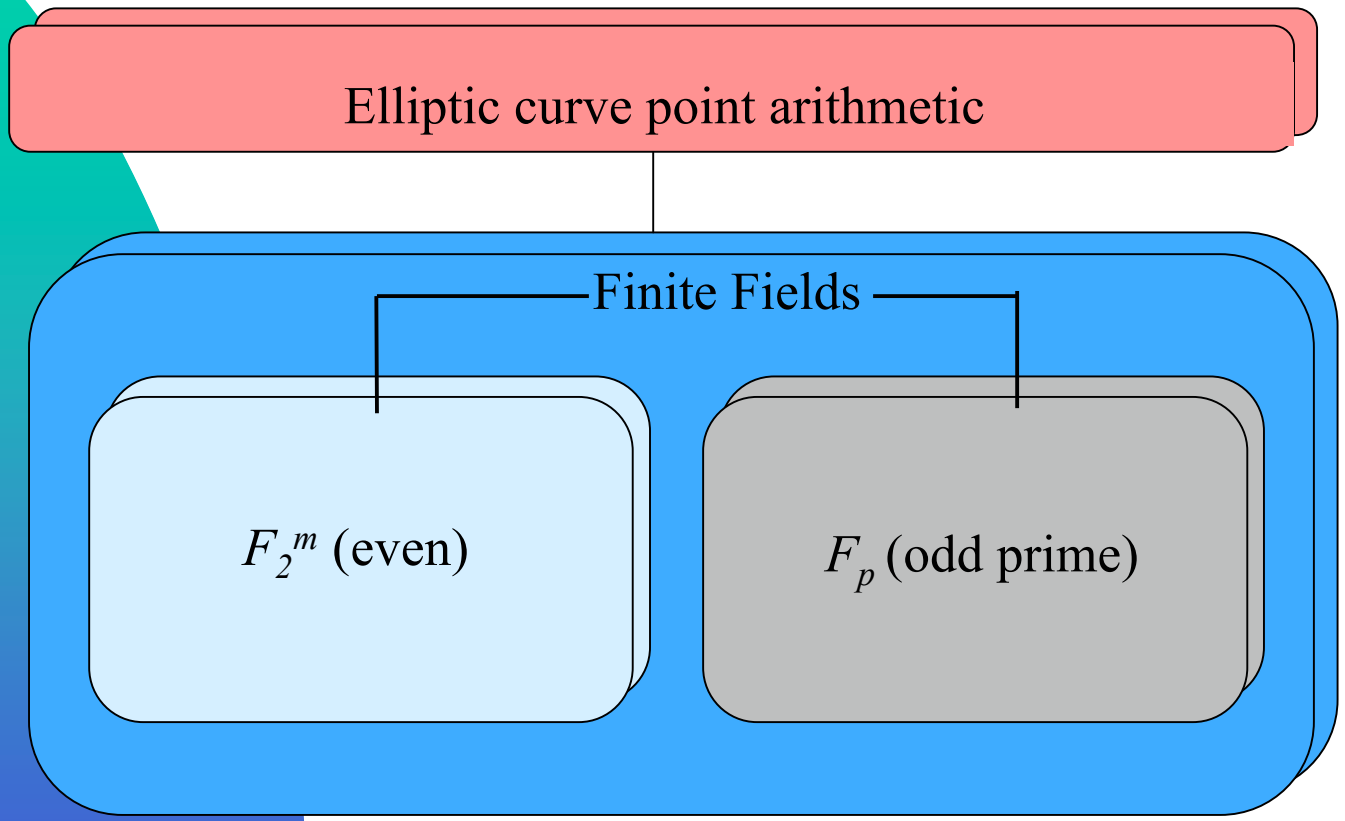
Elliptic curves and Finite fields

The fields *GF(p)* and $GF(2^m)$ can be used:

- *GF(p)*
  - Montgomery
  - Special primes : 2^k-c, 2^k-1 etc.
- $GF(2^m)$
  - Polynomial Basis, Normal Basis
  - Trinomials, all-one-polynomials
  - Composite Fields, Montgomery

  $GF(2^m)$ offers higher performance and less area consumption.

Elliptic curves over finite fields



CINVESTAV

Finite fields: definitions and operations

$F_2{}^m$ finite field operations : Addition, Squaring, multiplication and inversion

Francisco Rodríguez Henríquez

The field $F_2{}^m$

Let us consider a finite field $F=GF(2^m)$ over $K=GF(2)$.

Elements of F: Polynomials of degree less than m, with coefficients in K, such that,

$$\{a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\ldots+a_2x^2+a_1x+a_0 | a_i = 0 \text{ or } 1\}.$$

Fact: The field F has exactly $q-1=2^m-1$ nonzero elements plus the zero element.

Generating polynomial and polynomial basis

The finite field $F=GF(2^m)$ is completely described by a monic irreducible polynomial, often called generating polynomial, of the form

$$P(x) = x^m + k_{m-1}x^{m-1} + k_{m-2}x^{m-2} + \ldots + k_1 x + k_0$$

Where $k_i \in GF(2)$ for $i=0,1,\ldots,m-1$.

Let α be a primitive root of $P(x)$, i.e., $P(\alpha) = 0$. Then, we define the polynomial basis of $GF(2^m)$ over $GF(2)$ using the primitive element α and its m first powers

$$\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\},$$

which happen to be linearly independent over $GF(2)$.

Polynomial representation

Using the polynomial basis we can uniquely represent any number $A \in F=GF(2^m)$ as

$$A = \sum_{i=0}^{m-1} a_i \alpha^i$$

Sometimes, it is more convenient to represent a field element using the so-called coordinate representation,

Polynomial Rep.	$\leftrightarrow$	Coordinate Rep.
$a_{m-1}\alpha^{m-1} + a_{m-2}\alpha^{m-2} + \ldots + a_1\alpha + a_0$	$\leftrightarrow$	$(a_{m-1}, a_{m-2} \ldots, a_1, a_0)$

Example: Nonzero Elements of GF(2^4) with defining polynomial: $f(x)=x^4+x+1$.

CINVESTAV

i	α^i	Coordinates
0	1	(0 0 0 1)
1	α	(0 0 1 0)
2	α^2	(0 1 0 0)
3	α^3	(1 0 0 0)
4	$\alpha^4=\alpha+1$	(0 0 1 1)
5	$\alpha^5=\alpha^2+\alpha$	(0 1 1 0)
6	$\alpha^6=\alpha^3+\alpha^2$	(1 1 0 0)
7	$\alpha^7=\alpha^3+\alpha+1$	(1 0 1 1)
8	$\alpha^8=\alpha^2+1$	(0 1 0 1)
9	$\alpha^9=\alpha^3+\alpha$	(1 0 1 0)
10	$\alpha^{10}=\alpha^2+\alpha+1$	(0 1 1 1)
11	$\alpha^{11}=\alpha^3+\alpha^2+\alpha$	(1 1 1 0)
12	$\alpha^{12}=\alpha^3+\alpha^2+\alpha+1$	(1 1 1 1)
13	$\alpha^{13}=\alpha^3+\alpha^2+1$	(1 1 0 1)
14	$\alpha^{14}=\alpha^3+1$	(1 0 0 1)

Field Multiplication definition

Let $A, B \in F=GF(2^m)$ be two elements given in the polynomial basis as

$$A = \sum_{i=0}^{m-1} a_i \alpha^i \qquad \text{and} \qquad B(x) = \sum_{i=0}^{m-1} b_i x^i$$

The field product C' of the elements $A, B \in F$ is defined as,

$$C' = A(x)B(x) \bmod P(x)$$

where $P(x)$ is the generating polynomial.

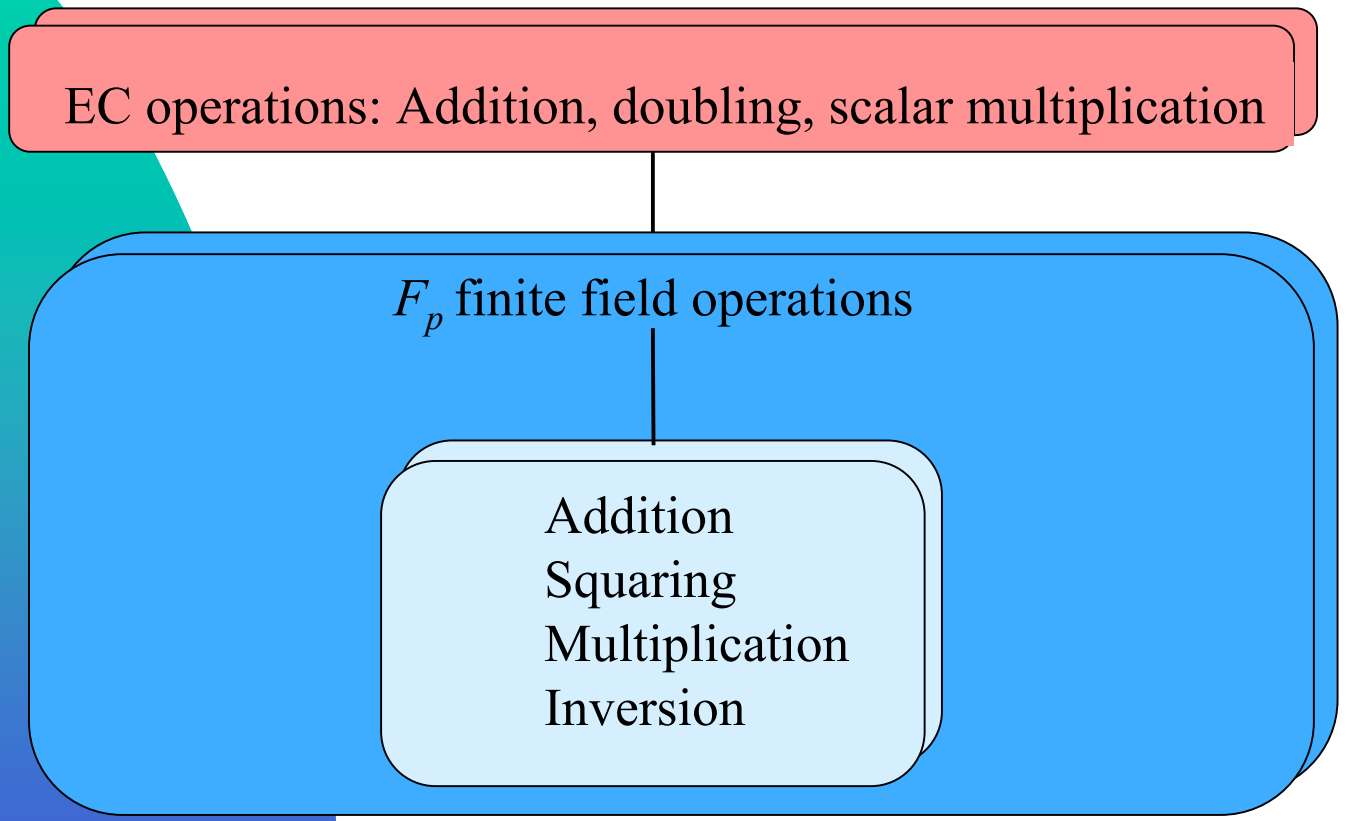
CINVESTAV

Elliptic Curves over GF(P)

Elliptic Curve operations: addition, doubling, scalar multiplication

Francisco Rodríguez Henríquez

Elliptic curves over finite fields



Elliptic Curves

CINVESTAV

"It is possible to write endlessly on elliptic curves (This is not a threat)"
Serge Lang, mathematician

- Elliptic curves as algebraic/geometric entities have been studied extensively for the past 150 years, and from these studies has emerged a rich and deep theory. Originally persued for purely aesthetic reasons, elliptic curves have recently been utilized in devising algorithms for factoring integers, primality tests, and in public-key cryptography.
- Elliptic curve systems as applied to cryptography were first proposed in 1985 independently by Neal Koblitz from the University of Washington, and Victor Miller, who was then at IBM, Yorktown Heights.

Elliptic Curves

- An elliptic curve over real numbers is defined as the set of points *(x,y)* which satisfy an elliptic curve equation of the form:

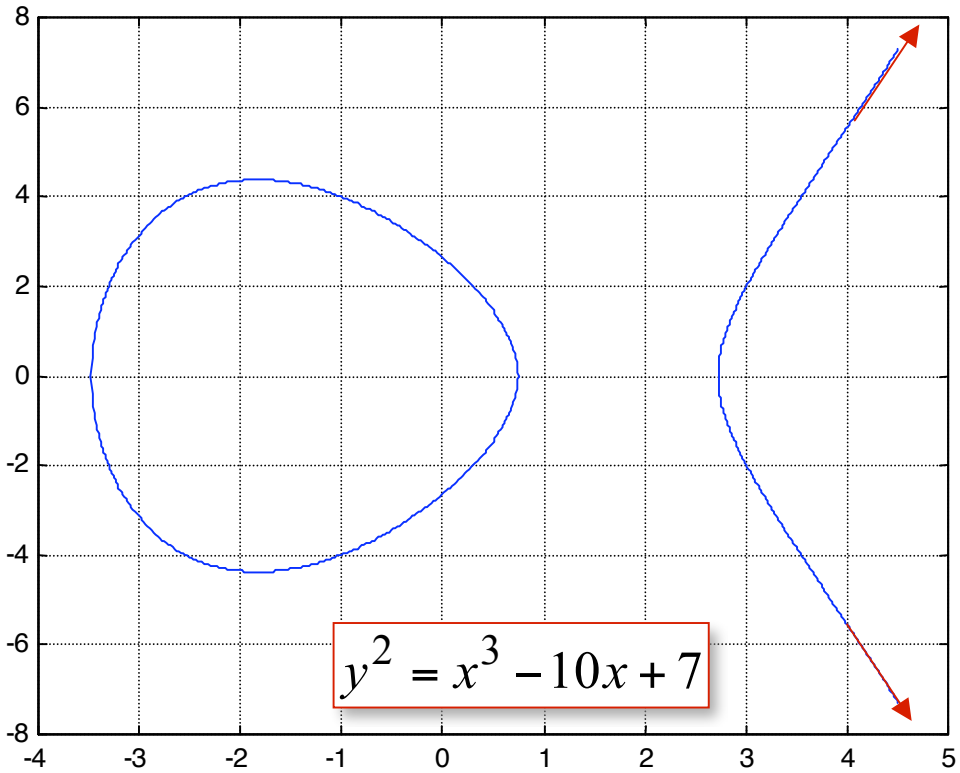
$$y^2 = x^3 + ax + b,$$

where x, y, a and b are real numbers,

- and the right side part of the equation, i.e., $x^3 + ax + b$ contains no repeated factors, or equivalently if

$$4a^3 + 27b^2 \neq 0$$

then the elliptic curve can be used to form a group.



Elliptic Curve Addition

1. $P + O = O + P = P$ for all $P \in E(Z_p)$.
2. If $P = (x, y) \in E(Z_p)$, then $(x, y) + (x, -y) = O$. (The point $(x, -y)$ is denoted by $-P$, and is called the *negative* of P; observe that $-P$ is indeed a point on the curve.)
3. Let $P = (x_1, y_1) \in E(Z_p)$ and $Q = (x_2, y_2) \in E(Z_p)$, where $P \neq -Q$. Then $P+Q = (x_3, y_3)$, where

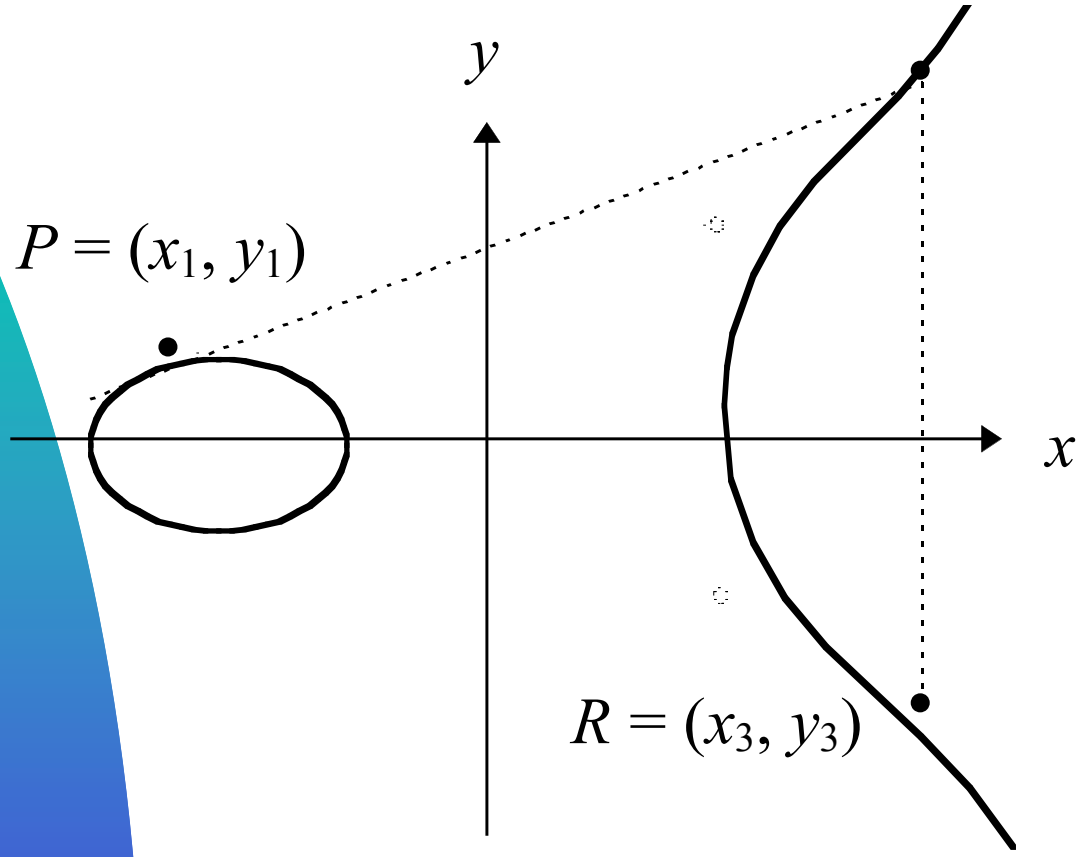
$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda (x_1 - x_3) - y_1,$$

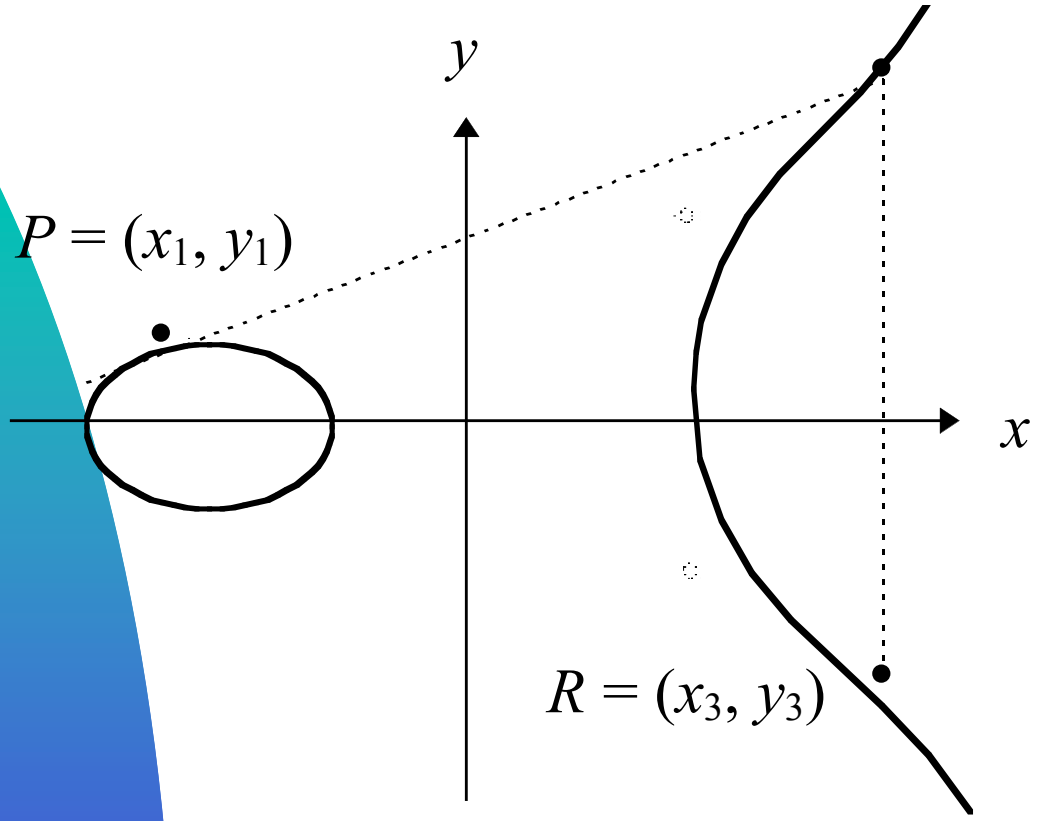
and

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q. \end{cases}$$

Elliptic Curve Addition



Elliptic Curve Addition



Elliptic Curve Addition

CINVESTAV

- An *elliptic curve* E over Zp is defined by an equation of the form
- $y^2 = x^3 + ax + b,$ (*)
- where $a, b \in Zp,$ and $4a^3 + 27b^2 \neq 0 \pmod{p}$, together with a special point O, called the *point at infinity*. The set $E(Zp)$ consists of all points (x, y), $x \in Zp$, $y \in Zp$, which satisfy the defining equation (*), together with O.

Elliptic Curve Addition

CINVESTAV

Example 1 *(elliptic curve over Z_{23}) Let $p = 23$* and consider the elliptic curve *E:* $y^2 = x^3 + x + 1$ defined over Z_{23}. *(*In the notation of equation *(*),* we have $a = 1$ and $b = 1$.*)* Note that $4a^3+27b^2 = 4 + 4 = 8 \neq 0$, so *E* is indeed an elliptic curve. The points in $E(Z_{23})$ are *O* and the following:

(0, 1)(0, 22)(1, 7)(1, 16)(3, 10)(3, 13)(4, 0)(5, 4)(5, 19)

(6, 4)(6, 19)(7, 11)(7, 12)(9, 7)(9, 16)(11, 3)(11, 20)

(12, 4)(12, 19)(13, 7)(13, 16)(17, 3)(17, 20)(18, 3)

(18, 20)(19, 5)(19, 18)

Elliptic Curve Addition

CINVESTAV

Example 2 *(elliptic curve addition)* Consider the elliptic curve defined in Example 1.

Let $P = (3, 10)$ *and* $Q = (9, 7)$. Then $P + Q = (x_3, y_3)$ is computed as follows:

$$\lambda = \frac{7 - 10}{9 - 3} = \frac{-3}{6} = \frac{-1}{2} = -2^{-1} = 11 \in \mathbb{Z}_{23}.$$

Note that $2^{-1} = 12$ since $2 \cdot 12 \equiv 1 \pmod{23}$. Finally,

$x3 = 112 - 3 - 9 = 6 - 3 - 9 = -6 \equiv 17 \pmod{23}$, and

$y3 = 11(3 - (-6)) - 10 = 11(9) - 10 = 89 \equiv 20 \pmod{23}$.

Hence $P + Q = (17, 20)$.

Elliptic Curve Addition

CINVESTAV

1. Let $P = (3,10)$. Then $2P = P + P = (x_3, y_3)$ is computed as follows:

$$\lambda = \frac{3(3^2) + 1}{20} = \frac{5}{20} = \frac{1}{4} = 4^{-1} = 6 \in \mathbb{Z}_{23}.$$

Note that $4\text{-}1 = 6$ since $4 \bullet 6 \equiv 1 \pmod{23}$. Finally,

$x_3 = 62 - 6 = 30 \equiv 7 \pmod{23}$, and

$y_3 = 6\,(3 - 7) - 10 = -24 - 10 = -34 \equiv 12 \pmod{23}$.

Hence $2P = (7,12)$.

Elliptic Curve Addition

CINVESTAV

For historical reasons, the group operation for an elliptic curve $E(Z_p)$ has been called *addition*. By contrast, the group operation in is *multiplication*. The differences in the resulting additive notation and multiplicative notation can sometimes be confusing. Table 1 shows the correspondence between notation used for the two groups and $E(Z_p)$.

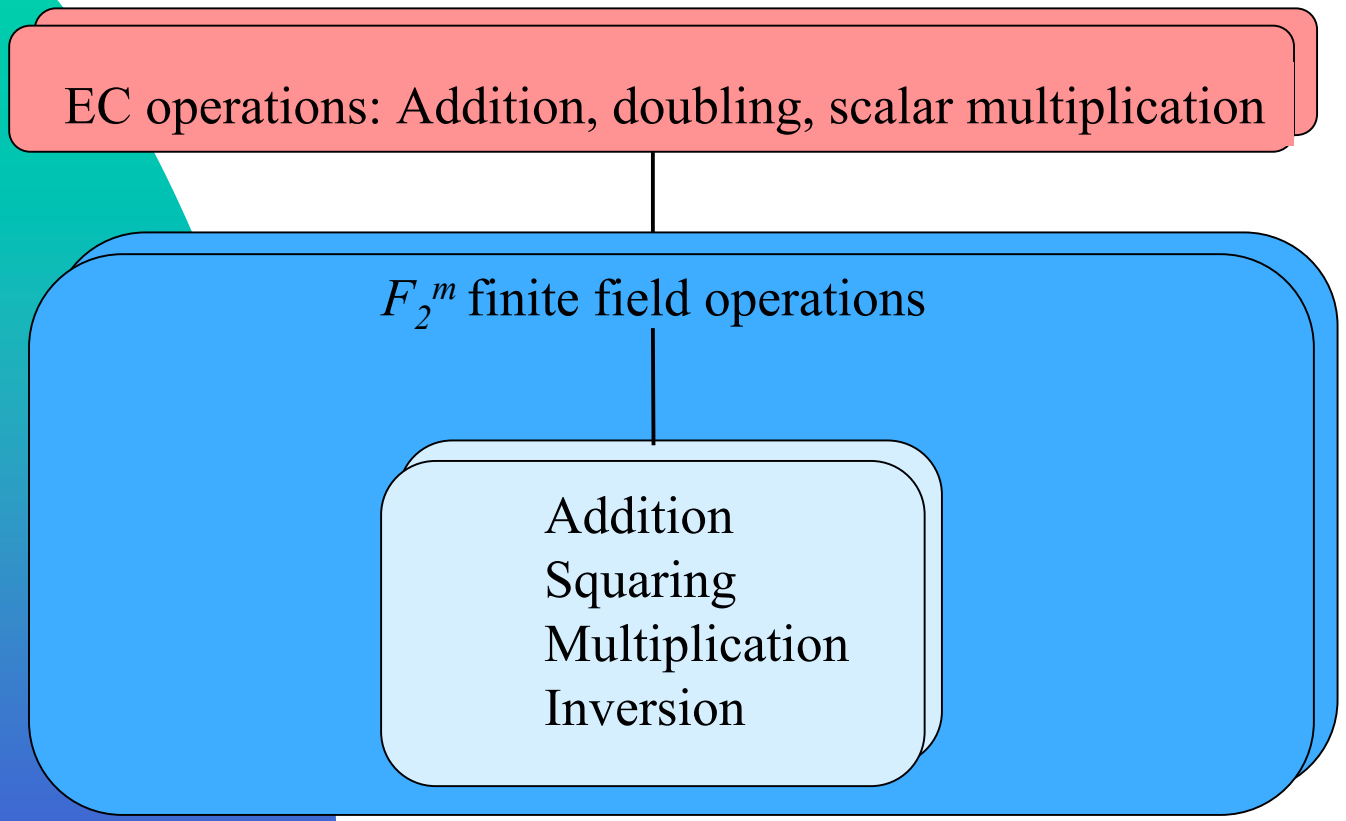
CINVESTAV

Elliptic Curves over GF(2^m)

Elliptic Curve operations: addition, doubling, scalar multiplication

Francisco Rodríguez Henríquez

Elliptic curves over finite fields



Binary elliptic curve definition

Let $F_q = GF(2^m)$ be a finite field of characteristic two. A non-supersingular elliptic curve $E(F_q)$ is defined to be the set of points $(x, y) \in GF(2^m) \times GF(2^m)$ that satisfy the equation,

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

Where a and b are in F_q, and $b \neq 0$, together with the point at infinity denoted by O.

Binary elliptic curves

CINVESTAV

The binary elliptic curve is defined as the collection or set of all points (x,y) which satisfy the elliptic curve equation over $F=GF(2^m)$ (where x and y are elements in the field F). An elliptic curve group over F consists of the points on the corresponding elliptic curve, together with a point at infinity, O. There are finitely many points on such an elliptic curve.

An Example

As a very small example, once again, consider the field F_{24}, defined by using polynomial representation with the irreducible polynomial

$$f(x) = x^4 + x + 1$$

Consider the elliptic curve

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

$$y^2 + xy = x^3 + \alpha^4 x + 1$$

The point (α^5, α^3) satisfies this equation over F_{24} ,

$$y^2 + xy = x^3 + \alpha^4 x + 1$$

$$\left(\alpha^3\right)^2 + \alpha^5\alpha^3 = \left(\alpha^5\right)^3 + \alpha^4\alpha^{10} + 1$$

$$\alpha^6 + \alpha^8 = \alpha^{15} + \alpha^{14} + 1$$

An Example

CINVESTAV

And by consulting the discrete log table for this field, we can see that indeed,

$$(1100)+(0101) = (0001)+(1001)+(0001)$$

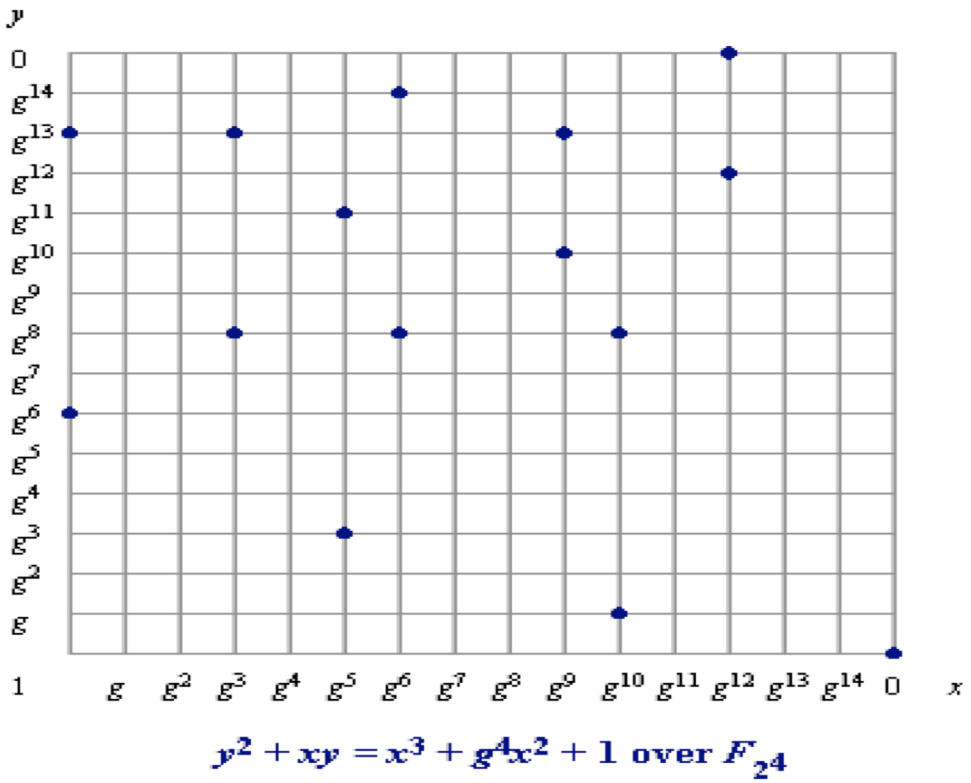
$$(1001) = (1001)$$

There exist a total of fifteen points which satisfy this equation, namely,

$$(1, \alpha^{13})\ (\alpha^{3}, \alpha^{13})\ (\alpha^{5}, \alpha^{11})\ (\alpha^{6}, \alpha^{14})\ (\alpha^{9}, \alpha^{13})\ (\alpha^{10}, \alpha^{8})\ (\alpha^{12}, \alpha^{12})$$

$$(1, \alpha^{6})\ (\alpha^{3}, \alpha^{8})\ (\alpha^{5}, \alpha^{3})\ (\alpha^{6}, \alpha^{8})\ (\alpha^{9}, \alpha^{10})\ (\alpha^{10}, \alpha)\ (\alpha^{12}, 0)\ (0, 1)$$

An Example



$y^2 + xy = x^3 + g^4x^2 + 1$ over F_{24}

Arithmetic on elliptic curves

CINVESTAV

The negative of the point $P=(x_P, y_P)$ in the elliptic curve is the point defined as

$$-P = (x_P, x_P + y_P).$$

Where we have the following property, $P+(-P)=O$, the point at infinity. Furthermore, $P+O=P$ for all points P in the elliptic curve group. Also, if $x_P=0$, then $2P=O$.

Arithmetic on Elliptic Curves

- Addition and Doubling
  - $P = (x_1 , y_1)$ and $Q = (x_2 , y_2)$, then $P + Q = (x_3 , y_3)$
  - $x_3 = \lambda^2 - x_1 - x_2$
  - $y_3 = \lambda(x_1 - x_3) - y_1$
- where
  - $\lambda = (y_2 - y_1) / (x_2 - x_1)$ for $P \neq Q$
  - $\lambda = (3x_1^2 + a) / 2y_1$ for $P = Q$ (doubling)
- Scalar Multiplication
  - $P=kP=P+P+\ldots+P$ ← k times

- Arithmetic performed in the finite field $F=GF(2^m)$ over $K=GF(2)$.
- Arithmetic on elliptic curves requires addition, squaring, multiplication and inversion in finite fields.

Scalar Multiplication and Order

- Elliptic curve points can be added but not multiplied. It is however, possible to perform *scalar multiplication*, which is another name for repeated addition of the same point.
- If n is a positive integer and P a point on an elliptic curve, the scalar multiple nP is the result of adding $n-1$ copies of P to itself.
- The total number of points in the curve, including the point O, is called the *order* of the curve.

Order notions in elliptic curves

CINVESTAV

The order of an elliptic Curve. Notice that there can only be a finite number of points on the curve. Even if every possible pair *(x,y)* were on the curve, there would be only p^2 or $(2^m)^2=2^{2m}$ possibilities. The total number of points, including the point *O*, is called the order of the elliptic curve. The order is written $\#E(F_q)$.

As a matter of fact, the curve $E(F_q)$ could have at most $2q+1$ points because we have one point at infinity and $2q$ pairs (x, y), because for each x, we have two values of y.

A celebrated result discovered by Hasse, gives the lower and upper bounds for the order of a curve. Let $\#E(F_q)$ be the number of point in $E(F_q)$. Then,

$$\left|\#E\left(F_q\right)-\left(q+1\right)\right| \leq 2\sqrt{q}$$

Order notions in elliptic curves

CINVESTAV

The order of a point.

As we did in the case of finite fields, we can also introduce the concept of the order of an element in elliptic curves. The order of a point P is the smallest integer k such that $kP = 0$. The order of any point is always defined, and divides the order of the curve $\# E(F_q)$. This guarantees that if r and l are integers, then $rP=lP$ iff r=l mod k.

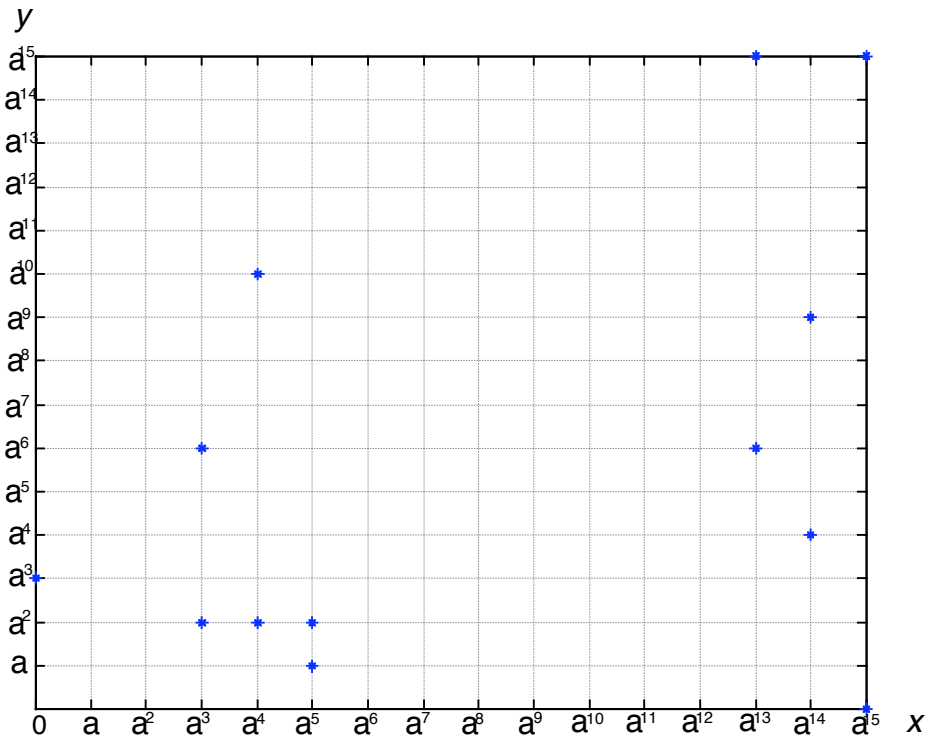
Elements in an elliptic curve

Example: Let $F=GF(2^4)$ be a binary finite field with defining polynomial: $f(x)=x^4+x+1$.

Consider the elliptic curve,

$$y^2 + xy = x^3 + \alpha^{13}x^2 + \alpha^6$$

There exist a total of 14 solutions in such a curve, including the point at infinite O.



CINVESTAV

Elliptic curve primitives

Elliptic curve primitives: Key-pair generation, Signing and Verification

Francisco Rodríguez Henríquez

Elliptic curve cryptosystem challenge

Elliptic Curve Discrete Logarithm Problem (ECDLP).

Problem: Given P, Q in $E(F_q)$, defined in F_{2m} as,

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

with $ord(P)=n$.

Find an integer m with $1 \leq k \leq n-1$, such that,

$$Q=kP$$

Some comments on ECC

CINVESTAV

Cryptographical motivation

Public Key Cryptosystems: what key size?

Maximal number of operations, in a reasonable amount of time: 2^{50}–2^{60}.

Secure means: unbreakable with less than 2^{80} operations.

1. for any system, key size > 80 bits. (Try all keys)
2. for a system based on Dlog in a group, key size > 160 bits. (Pollard's Rho)
3. for a system based on factorization, key size > 1024 bits. (NFS)

Some comments on ECC

Elliptic curves. "optimal" for Dlog based cryptosystems:

- Group order can be computed quickly (Schoof–Elkies–Atkin, Satoh–ECPC).
- In general, the best known attack is Pollard's Rho.

Elliptic Curve Cryptography

CINVESTAV

ECDSA key generation. Each entity A does the following:

1. Select an elliptic curve E defined over Z_p. The number of points in $E(Z_p)$ should be divisible by a large prime n.
2. Select a point $P \in E(Z_p)$ of order n.
3. Select a statistically unique and unpredictable integer d in the interval $[1, n - 1]$.
4. Compute $Q = dP$.
5. *A's* public key is (E, P, n, Q); *A's* private key is d.

Elliptic Curve Cryptography

ECDSA signature generation. To sign a message m, A does the following:

Select a statistically unique and unpredictable integer k in the interval $[1, n - 1]$.

Compute $kP = (x1, y1)$ and $r = x1 \bmod n$. (Here $x1$ is regarded as an integer, for example by conversion from its binary representation.)

If $r = 0$, then go to step 1. (This is a security condition: if $r = 0$, then the signing equation $s = k\text{-}1\{h(m) + dr\} \bmod n$ does not involve the private key d!)

Compute $k\text{-}1 \bmod n$.

Compute $s = k\text{-}1\{h(m) + dr\} \bmod n$, where h is the Secure Hash Algorithm (SHA-1).

If $s = 0$, then go to step 1. (If $s = 0$, then $s\text{-}1 \bmod n$ does not exist; $s\text{-}1$ is required in step 2 of signature verification.)

The signature for the message m *is* the pair of integers (r, s).

Elliptic Curve Cryptography

CINVESTAV

ECDSA signature verification. To verify A's signature (r, s) on m, B should:

Obtain an authentic copy of A's public key (E, P, n, Q).

Verify that r and s are integers in the interval $[1, n - 1]$.

Compute $w = s^{-1} \bmod n$ and $h(m)$.

Compute $u_1 = h(m)w \bmod n$ and $u_2 = rw \bmod n$.

Compute $u_1P + u_2Q = (x_0, y_0)$ and $v = x_0 \bmod n$.

Accept the signature if and only if $v = r$.

An example of the discrete logarithm problem

Example: Let $F=GF(2^4)$ be a binary finite field with defining polynomial: $f(x)=x^4+x+1$.

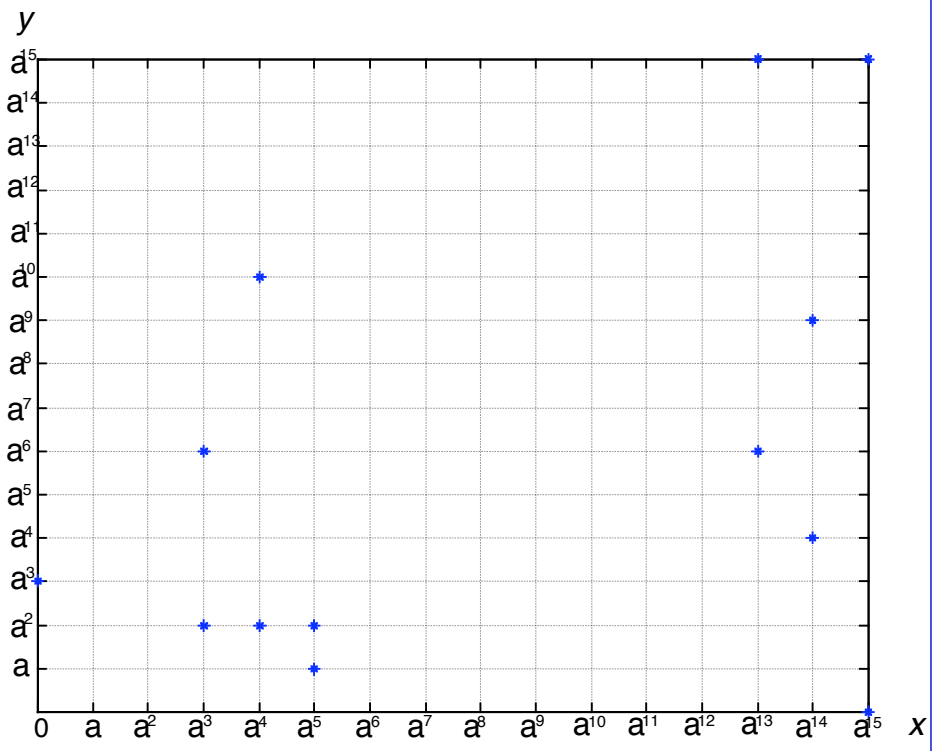
Consider again the elliptic curve,

$$y^2 + xy = x^3 + \alpha^{13}x^2 + \alpha^6$$

Given $Q = (\alpha^3, \alpha^6) = kP = k(\alpha^3, \alpha^2)$

Find k.

Use the following scalar table,



P	$2P$	$3P$	$4P$	$5P$	$6P$
(α^3, α^2)	(α^{13}, α^6)	(α^{14}, α^9)	(α^{14}, α^4)	$(\alpha^{13}, \alpha^{15})$	(α^3, α^6)

Order notions in elliptic curves

CINVESTAV

A Point of Prime Order

Given the elliptic curve, E, defined over the finite field, F_q , we want to fix a special point that will be used to mask the private key in a public/private key pair. The properties of $\boldsymbol{P}$ are important to the security of our system. Not just any point will do: we need a point $\boldsymbol{P}$, whose order n is prime; the larger the prime is , the more secure the cryptosystem is. Remember that $\boldsymbol{P}$ is of the form $\boldsymbol{P}=(x,y)$ where x and y satisfy the elliptic curve equation. Usually, we write x and y as x_P and y_P. Therefore, the special point $\boldsymbol{P}$ gives us two parameters:

- A point $\boldsymbol{P}=(x_P, y_P)$ of prime order
- The order n of $\boldsymbol{P}$

$\boldsymbol{P}$ is sometimes called the base point.

Elliptic curve parameters

Notation	Name	Description
F_q	base field	Either: F_p: $\{0,1,...,p-1\}$ with arithmetic *mod* p or, F_{2^m}
a, b	coefficients of the curve	a and b are elements of F_q . They determine an equation, which depends on the base field: For F_p: $y^2=x^3+ax+b$ For F_{2^m} $^2+xy=x^3+ax^2+b$
$\boldsymbol{P}$	point of prime order or base point	(x_P, y_P) The pair x_P, y_P satisfies the curve equation.
n	order of $\boldsymbol{P}$	The smallest nonzero number such that $\boldsymbol{P}$ added to itself n times, is the zero point, O, on the curve, n is prime.

Elliptic Curve Key Pair Generation

Elliptic curve parameters can be used to generate a public/private key pair. Elliptic curve parameters can either be common to several key pairs or specific to one key pair. The elliptic curve parameters can be public; the security of the system does not rely on these parameters being secret.

Creating the Key Pair

To compute a public/private key pair:

1. Generate a random value, d, between 1 and $n-1$.

2. Compute the elliptic curve point $d\boldsymbol{P}$, that is, $\boldsymbol{P}$ added to itself d times. Call this point $\boldsymbol{Q}$; it is a pair of field elements (x_Q, y_Q).

The key pair is $(\boldsymbol{Q}, d)$: $\boldsymbol{Q}$ is the public key, d is the private key. As mentioned before, even if you know $\boldsymbol{P}$ and $\boldsymbol{Q}$, you cannot easily calculate d.

ECDSA Signature Scheme

CINVESTAV

Signing a Message

The holder of the private key can sign a message as follows:

1. Digest the outgoing message using SHA1. This yields a *20*-byte (*160*-bit) digest, *e*.

2. Compute a random value, *k*, between *1* and *n–1*.

3. Compute the elliptic curve point $k\boldsymbol{P}=(x_1, y_1)$.

4. Currently, the first coordinate, x_1, is an element of the finite field. Set $r= x_1$.

5. Compute $s=k^{-1}(e+dr) \bmod n$, and check that *s* is nonzero.

The signature for this message is the pair *r* and *s*. Notice that, as with DSA, the signature depends on both the message and the private key. This means no one can substitute a different message for the same signature.

Note: The above equation is merely an outline. For cryptographic purposes, it is necessary to verify that certain numbers are nonzero, or that they satisfy other conditions.

Verifying a Signature

When a message is received, the recipient can verify the signature using the received signature values and the signer's public key, $\boldsymbol{Q}$. Because the pair (r,s) that has been received may not actually be a valid signature pair, it is customary to call the received pair (r', s') instead.

To verify a signature:

1. First verify that r' and s' are between 1 and $n-1$. If they are not, the output is invalid.

2. Digest the received message using SHA1. This yields a 20-byte (160-bit) digest, e.

3. Compute $c=(s')^{-1}$. Remember, s' is an integer $mod\ n$, so its inverse is also an integer $mod\ n$.

4. Compute $u_1=ec\ mod\ n$ and $u_2=r'c\ mod\ n$.

5. Compute the elliptic curve point $(x_1, y_1)=u_1\boldsymbol{P}+u_2\boldsymbol{Q}$.

6. Compute $v=x_1\ mod\ n$

If $v=r'$, the signature is verified. If they are different, the signature is invalid.

Verifying a Signature

The ECDSA algorithm depends in part on the fact that

$$\text{If } r=r' \bmod n, \text{ then } r\boldsymbol{P}=r'\boldsymbol{P}.$$

The following calculations are really just a series of substitutions that can be made by looking back at the definitions given in the previous sections. If the message has been signed correctly, then $s=s'$. Expanding the elliptic curve point

$$(x_1, y_1)=u_1\boldsymbol{P}+u_2\boldsymbol{Q}$$

calculated by the recipient, we see that:

$$u_1\boldsymbol{P}+u_2\boldsymbol{Q} = es^{-1}\boldsymbol{P}+rs^{-1}\boldsymbol{Q}=s^{-1}(e\boldsymbol{P}+r\boldsymbol{Q})$$

Recall that $\boldsymbol{Q}=d\boldsymbol{P}$, so:

$$u_1\boldsymbol{P}+u_2\boldsymbol{Q}=s^{-1}(e\boldsymbol{P}+r\boldsymbol{Q})=s^{-1}(e\boldsymbol{P}+rd\boldsymbol{P})=s^{-1}(e+dr)\boldsymbol{P}$$

Now recall that $s=k^{-1}(e+dr) \bmod n$, so:

$$\begin{aligned} u_1\boldsymbol{P}+u_2\boldsymbol{Q} &= s^{-1}(e+dr)\boldsymbol{P}=[k^{-1}(e+dr)]^{-1}(e+dr)\boldsymbol{P} \\ &= (k^{-1})^{-1}(e+dr)^{-1}(e+dr)\boldsymbol{P} \\ &= kP \end{aligned}$$

This is the point calculated by the recipient. But this is also the point generated by the sender. The recipient then checks that the information received was correct.

ECC and ECDSA: Differences

CINVESTAV

1. The private key d and the per-signature value k in ECDSA are defined to be *statistically unique and unpredictable* rather than merely *random* as in DSA. This is an important clarification and is a better statement of the security requirements. If k can be determined or if k repeats then an adversary can recover d, the private key. Of course, the use of a random value is explicitly stated as being allowed; however architecturally it is preferable to state the requirements rather than mandate a particular way to meet the requirements. For example, giving the requirements allows a high security implementation to filter the k values to ensure there are no repeats. This possibility is not allowed if k is required to be random. Also, stating the requirements gives more guidance to implementers and users regarding what constitutes a security concern.

ECC and ECDSA: Differences

CINVESTAV

2. In ECDSA, a method called point compression allows for a point on the elliptic curve (e.g., a public key Q) to be compactly represented by one field element and one additional bit, rather than two field elements. Thus, for example, if $p \approx 2160$ (so elements in Zp are 160-bit strings), then public keys can be represented as 161-bit strings. This can lead to a substantial reduction in the size of a public-key certificate, on the order of 25% when compared with other asymmetric algorithms.

ECC and ECDSA: Differences

CINVESTAV

2. In ECDSA, a method called point compression allows for a point on the elliptic curve (e.g., a public key Q) to be compactly represented by one field element and one additional bit, rather than two field elements. Thus, for example, if $p \approx 2160$ (so elements in Zp are 160-bit strings), then public keys can be represented as 161-bit strings. This can lead to a substantial reduction in the size of a public-key certificate, on the order of 25% when compared with other asymmetric algorithms.

Security and Efficiency

- **Security**
  1024-bit RSA and DSA ⟺ 160-bit ECC
  offers similar levels of security
- **Efficiency**
  - Computational overheads
  - ECC has shorter system parameters, keys, signatures
  - ECC is bandwidth efficient

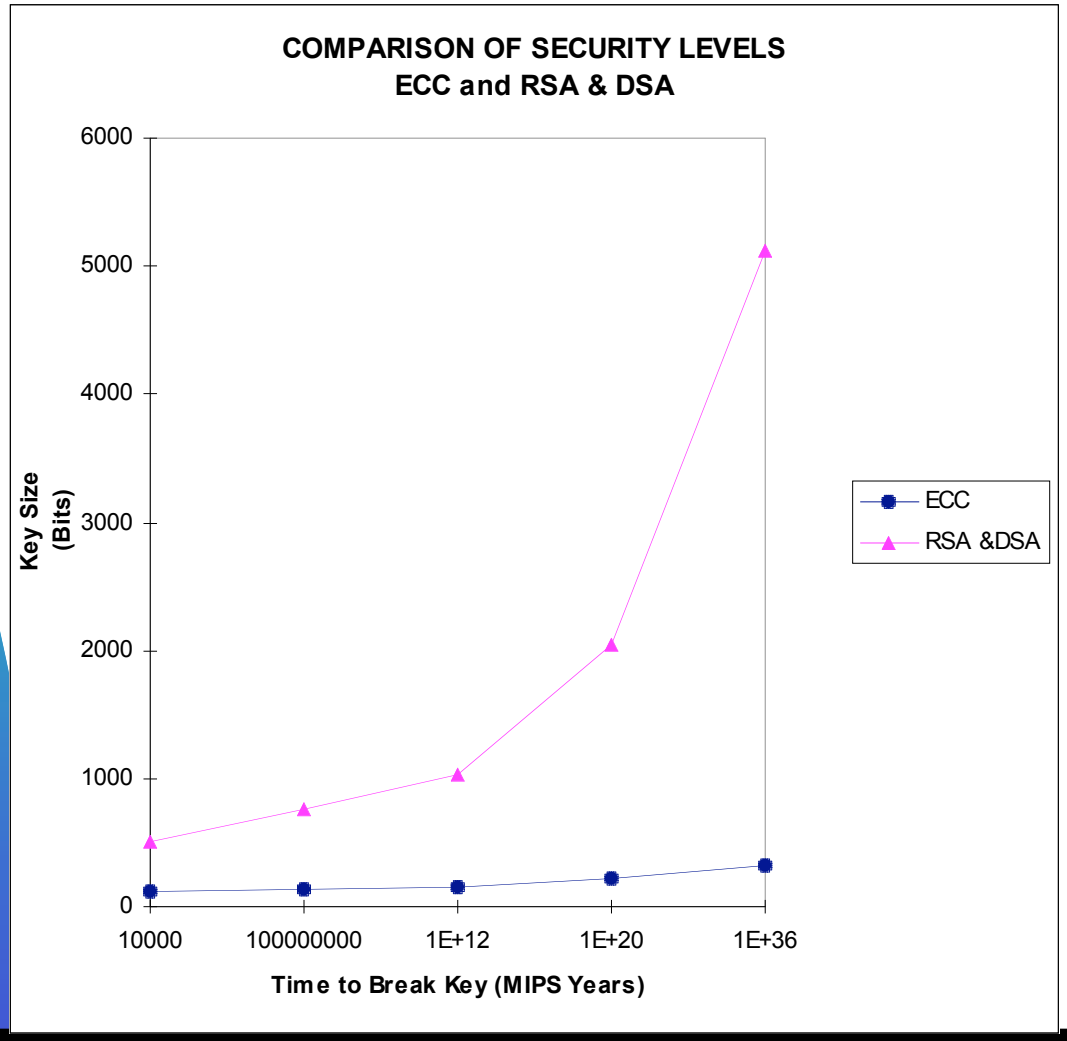
Some comments on ECC

Field size (in bits)	Size of n (in bits)	$\sqrt{\pi n / 2}$	MIPS years
163	160	2^{80}	9.6×10^{11}
191	186	2^{93}	7.9×10^{15}
239	234	2^{117}	1.6×10^{23}
359	354	2^{177}	1.5×10^{41}
431	426	2^{213}	1.0×10^{52}

Some comments on ECC

Size of n (in bits)	MIPS years
512	3×10^4
768	2×10^8
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

Some comments on ECC



ECC Applications

- ECC affords more efficient implementations than other public-key systems due to its extra strength:

  Storage efficiencies

  Bandwidth

  Computational efficiencies

- These lead to higher speeds, lower power consumption, and code size reductions.

- Therefore, it is very important for some applications such as:

  Wireless transactions, ATMs, cellular phones, storage of medical records, electronic cash, handheld computing, broadcast, and smart card applications.

EC Diffie-Hellman

- **User**

❶ *Choose* d_u *in [2, n-2]*

❷ $Q_u = d_u \times P$

❸ *Send* Q_u →

❹ ← *Receive* Q_s

❺ $K = d_u \times Q_s = d_u d_s \times P$

- **Server**

❶ *Choose* d_s *in [2, n-2]*

❷ $Q_s = d_s \times P$

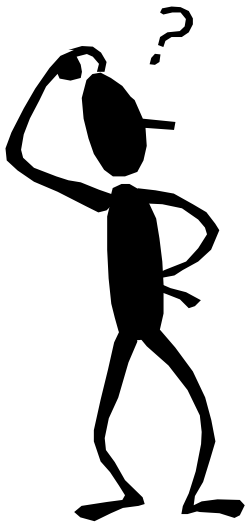
❸ *Receive* Q_u

❹ *Send* Q_s

❺ $K = d_s \times Q_u = d_s d_u \times P$

Implementation Considerations

- Computational overhead
- Power consumption
- Hardware complexity
- Computation delay
- Storage
- The number of messages exchanged



Scalar Multiplication

Point Multiplication *kP*

Core operation of the Elliptic Curve cryptography

Two approaches utilized:

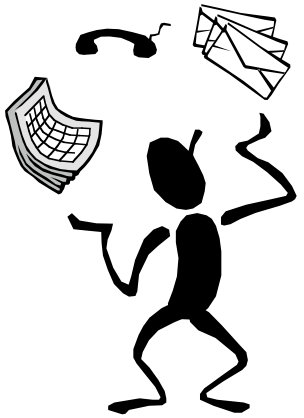
- Montgomery point multiplication with projective
- coordinates
- Double-and-add with P1363's projective coordinates

Scalar Multiplication

Operation	Montgomery Point Mult.	Double-and-Add (avg.)
# of Mults.	5m	7m
# of Sqrs.	6m	10.5m
# of Invrs.	1	1

ISL Implementation results

- We have implemented the followings in GF(2^{176}):
  - Field addition, multiplication, inversion and EC point doubling, addition, and multiplication operations…
- Tool : Microsoft Visual C++ Version 5.0
- Platform : PC with the 300-MHz Pentium II processor, running Windows NT 4.0
- The next table shows the timings for these operations



ISL Implementation results

Operation	Our Method (300- MHz P-II)	Reproduced (300-MHz P-II)	Original (133-MHz P-I)
Field Multiplication	12 μsec	15 μsec	(62.7+1.8)μsec
Field Squaring	1.5 μsec	2.5 μsec	(5.9+1.8) μsec
Field Inversion	60 μsec	63 μsec	160 μsec
EC Addition	80 μsec	83 μsec	306 μsec
EC Doubling	80 μsec	85 μsec	309 μsec
EC Multiplication	25 msec	30 msec	72 msec

ISL Implementation results

- ECC library was created for GF(p)
  - Curve generation
  - Curve operations, field operations
  - Sign and verify messages
  - Scalable

- 32-bit ARM software development kit is used
  - Processor : ARM7TDMI (32-bit RISC)
    - shortest instruction time : 800ns (at f = 80MHz)
    - 30 general purpose + 6 status registers
    - 48 instructions
    - optimized for the best combination of die size, performance and power consumption.
    - uses three stage pipeline : fetch, decode and execute.

ISL Implementation Results

Parameters	160-bit ECC	176-bit ECC	192-bit ECC	208-bit ECC	256-bit ECC
Bandwidth	1730	1826	1922	2018	2306
User Side Storage	1408	1520	1632	1744	2080

The protocol bandwidth and storage requirements in bits.

Some final comments on ECC

CINVESTAV

- **Is there anything better than RSA?**

  Yes: ECC

  (for the well known reasons mentioned before)

Some final comments on ECC

- **How long will we survive with ECC??**

- „the principle of hope“:
for the time being nobody expects a dramatic breakthrough

- in real life implementations we try to introduce an extra margin of security