Introduction to ECC

Nigel Smart

nigel@cs.bris.ac.uk

January 17, 2013

Why the need for Elliptic Curves?

Discrete logarithm problem example

Let $p =$
1053546280395016975304616582933958731948871814925913489342608734258717883575185867300386287737705577937382925873762451990450430661350859682697410256268271147283034897563214300237166369174066615907176472549470083113107138189921280884003892629359

NB: $p = 158(2^{800} + 25) + 1$ and has 807 bits.

Problem:
Find $\lambda \in \mathbb{Z}$ such that

$$2 \equiv 3^{\lambda} \pmod{p}.$$

Discrete logarithm problem

Let p and L be large primes such that $L | (p-1)$.

The multplicative group of integers modulo p contains an element g of order L.

The discrete logarithm problem:
Suppose $h \in \mathbb{Z}_p^*$ also has order L.
Find $\lambda \in \mathbb{Z}$ such that
$$h \equiv g^\lambda \pmod{p}.$$

One-way function:
Fast to compute g^λ but difficult to compute λ.

Generalisation of DLOGs

Can take any (finite) group.

Bad Choices:

- Additive group $\mathbb{Z}$ or $\mathbb{F}_q$.
- Multiplicative group of or $\mathbb{C}$.

Apparently Good Choices:

- Finite fields $\mathbb{F}_q^*$.
- Elliptic curves over finite fields.
- Ideal class groups of number fields.
- Jacobian varieties of curves over finite fields.

Subexponential Algorithms

For factoring and the discrete logarithm problem in finite fields $\mathbb{F}_q^*$ there are index calculus algorithms.

These have subexponential complexity

$$O(\exp(c(\ln N)^{1/3}(\ln \ln N)^{2/3})).$$

For solving the discrete logarithm problem in class groups and Jacobians of curves of sufficiently high genus there are index calculus algorithms of subexponential complexity

$$O(\exp(c(\ln N)^{1/2}(\ln \ln N)^{1/2})).$$

But elliptic curve groups generally have exponential complexity.

Basics on Elliptic Curves

Elliptic Curves

An elliptic curve over a field K is non-singular curve

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

with $a_1, a_2, a_3, a_4, a_6 \in K$.

From these constants we define

$$\begin{aligned}
b_2 &= a_1^2 + 4a_2, \\
b_4 &= a_1a_3 + 2a_4, \\
b_6 &= a_3^2 + 4a_6, \\
b_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2, \\
c_4 &= b_2^2 - 24b_4, \\
c_6 &= -b_2^3 + 36b_2b_4 - 216b_6.
\end{aligned}$$

Elliptic Curves

A curve is called non-singular if it has no singularities.

- Essentially the “curve” does not cross or intersect itself.

This is easy to detect since the “discrimiannt” Δ will be zero if the curve is singular

$$\Delta = -b_2^2 b_8 - 8b_4^3 - 27b_6^2 + 9b_2 b_4 b_6.$$

Think of this as related to the discriminant of a polynomial, which is zero when the polynomial has repeated roots.

The curve is considered to be the set of solutions to the equations, plus

- An additional special point at infinity $\mathcal{O}_E$.

This is considered to lie infinitely far up the y-axis.

Elliptic Curves

Two curves E and E' are isomorphic over K if there is a bi-rational map between them which preserves the point at infinity.

Two curves with variables X, Y and X', Y' are isomorphic over K if there are constants $r, s, t \in K$ and $u \in K^*$, such that the change of variables

$$X = u^2 X' + r\ , \quad Y = u^3 Y' + s u^2 X' + t \qquad (1)$$

transforms E into E'.

Two most used cases (in classical ECC) are

- Characteristic p : $K = \mathbb{F}_p$, p a large prime
- Characteristic 2 : $K = \mathbb{F}_{2^n}$.

In pairing based crypto we also use

- Characteristic p : $K = \mathbb{F}_{p^n}$ for small n.

Elliptic Curves: Char p

In char p all curves are isomorphic to one of the form

$$E_{A,B} : Y^2 = X^3 + AX + B,$$

in which case we have

$$\Delta = -64A^3 - 432B^2.$$

Two curves in this form $E_{A,B}$ and $E_{A',B'}$ are isomorphic over K if

$$A' = u^4 A \text{ and } B' = u^6 B \text{ for } u \in K^*.$$

Elliptic Curves: Char p

If $-3/A$ is a fourth root in K^* then we can replace $E_{A,B}$ by the curve

$$E_{-3,B'} : Y^2 = X^3 - 3X + B',$$

which will provide a lot of efficiency gains later.

- In practice it is rare to choose $A \neq -3$ for classical cryptography.

The value $-3/A$ will be a fourth root

- 25 percent of the time when $p \equiv 1 \pmod 4$.
- 50 percent of the time when $p \equiv 3 \pmod 4$.

Elliptic Curves: Char 2

In char 2 all curves are isomorphic to one of the form

$$E_{A,B} : Y^2 + XY = X^3 + AX^2 + B,$$

where $A \in \{0, \gamma\}$ where γ is a fixed element in B of trace one.

In which case we have

$$\Delta = B.$$

Note for later, the number of elements in $E_{A,B}(K)$ is divisible by 4 if the trace of A is zero, and divisible by 2 otherwise.

Since we want curves with small cofactor and we usually choose fields of odd exponent, e.g. $K = \mathbb{F}_{2^p}$ where p is prime it is common to select

$$A = 1,$$

since this aids efficiency.

The Group Law

Elliptic Curves As Groups

$E(K) = \{\text{points } (x, y) \in K^2\} \cup \{\mathcal{O}_E\}$.

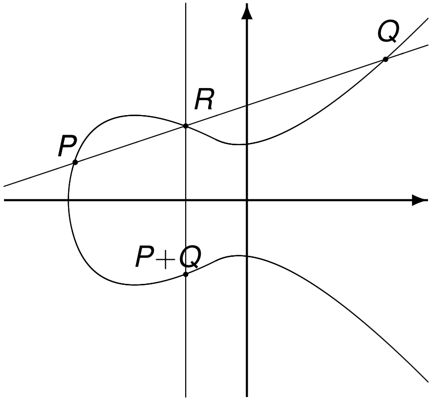
Point Addition

There is a process which, given two points (x_1, y_1) and (x_2, y_2), gives a third point (x_3, y_3).

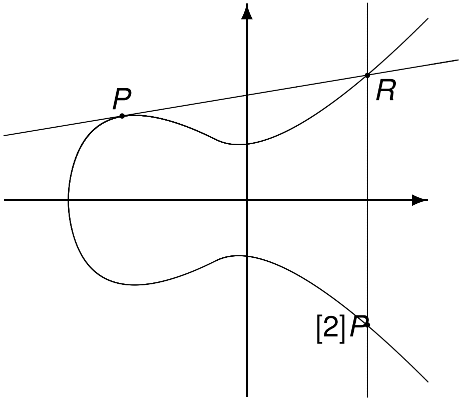
This addition process makes the set $E(K)$ an Abelian group with identity $\mathcal{O}_E$.

- An Abelian group is what you need for a lot of crypto protocols.

Adding two points on an elliptic curve



Doubling a point on an elliptic curve



Addition Formulae

We can write down formulae for the addition law

- Hence, can compute with the addition law

This can be done with the general equation in any characteristic.

We shall give the simplifications in the two main cases.

Addition Formulae: Char p

Suppose we are in characteristic p

$$E : Y^2 = X^3 + AX + B$$

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on E.

Negation in group law is given by

- $-P_1 = (x_1, -y_1)$.

Addition Formulae: Char p

Suppose

$$P_3 = (x_3, y_3) = P_1 + P_2$$

then

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= (x_1 - x_3)\lambda - y_1. \end{aligned}$$

where when $x_1 \neq x_2$ we set

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1},$$

and when $x_1 = x_2$ and $y_1 \neq 0$ we set

$$\lambda = \frac{3x_1^2 + A}{2y_1}.$$

Addition Formulae: Char 2

Suppose we are in characteristic 2

$$E : Y^2 + XY = X^3 + AX^2 + B$$

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on E.

Negation in group law is given by

- $-P_1 = (x_1, y_1 + x_1)$.

Addition Formulae: Char 2

Suppose

$$P_3 = (x_3, y_3) = P_1 + P_2$$

then

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + A + x_1 + x_2, \\ y_3 &= (x_1 + x_3)\lambda + x_3 + y_1. \end{aligned}$$

where when $x_1 \neq x_2$ we set

$$\lambda = \frac{y_2 + y_1}{x_2 + x_1},$$

and when $x_1 = x_2 \neq 0$ we set

$$\lambda = \frac{x_1^2 + y_1}{x_1}.$$

Cost of Addition Formulae: Char p

Point Addition:

- 6 Field Additions (Trivial)
- 3 General Field Multiplications
- 1 Field Inversion

Point Doubling:

- 5 Field Additions (Trivial)
- 2 Scalar/Field Multiplications (Trivial)
- 4 General Field Multiplications
- 1 Field Inversion

Cost of Addition Formulae: Char 2

Point Addition:

- 9 Field Additions (Trivial)
- 1 Field squaring (Trivial in Char 2)
- 2 General Field Multiplications
- 1 Field Inversion

Point Doubling:

- 8 Field Additions (Trivial)
- 2 Field squaring (Trivial in Char 2)
- 2 General Field Multiplications
- 1 Field Inversion

Note point doubling requires fewer multiplications than in the char p case.

The ECDLP

Given $P = (x, y)$ and an integer n we can efficiently compute

$$nP = \underbrace{(x, y) + (x, y) + \cdots + (x, y)}_{n \text{ times}}$$

using the double-and-add method.

The Order of the point P is the smallest number $L > 0$ such that

$$LP = \mathcal{O}_E.$$

Assume that L is a ‘large’ prime.

ECDLP

Suppose that $Q = (x', y')$ is some other point of order L.

Then there is some number λ such that

$$Q = \lambda P.$$

The ECDLP is to find this number λ.

- This problem is believed to be hard.
- This gives a one way function.

It is believed in general that the best algorithm to solve this problem takes time

$$O(\sqrt{L}).$$

i.e. fully exponential complexity.

Choices of Finite Field Arithmetic

Recap on Finite Fields

A Field is a set with two operations $(G, \times, +)$ such that

- $(G, +)$ is an abelian group, identity denoted by 0.
- $(G \setminus \{0\}, \times)$ is an abelian group
- $(G, \times, +)$ satisfies the distributive law

Distributive law

For all $f, g, h \in (G, \times, +)$

$$f \times (g + h) = (f \times g) + (f \times h).$$

Examples

Rational numbers, real numbers, complex numbers, integers modulo p.

Fields

We define the set of invertible elements of $\mathbb{Z}/N\mathbb{Z}$ as

$$(\mathbb{Z}/N\mathbb{Z})^* = \{a \in \mathbb{Z}/N\mathbb{Z} : \gcd(a, N) = 1\}.$$

The set $(\mathbb{Z}/N\mathbb{Z})^*$ is always a group with respect to multiplication and clearly has size $\phi(N)$.
When N is a prime p we have

$$\mathbb{Z}/N\mathbb{Z}^* = \{1, \ldots, p - 1\}.$$

We define the sets

$$\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z} = \{0, \ldots, p - 1\} \quad \text{and} \quad \mathbb{F}_p^* = (\mathbb{Z}/p\mathbb{Z})^* = \{1, \ldots, p - 1\}.$$

We call $\mathbb{F}_p$ a finite field of characteristic p.
Finite fields are of central importance in coding theory and cryptography.

Finite Field Arithmetic

It is crucial to have a good finite field arithmetic.

A number of different choices have been proposed.

We shall now recap on the main two choices.

- $\mathbb{F}_{2^p}$, p prime
- $\mathbb{F}_p$, p prime

Characteristic Two Fields

Characteristic Two Fields

Of particular interest are fields of char 2.

Take an irreducible binary polynomial f of degree n and let $\mathbb{F}_{2^n}$ denote all the binary polynomials of degree $< n$.

Addition in $\mathbb{F}_{2^n}$ is defined as

- $a \oplus b = a + b \pmod{2}$
- Note this means $-a = a$.

Multiplication in $\mathbb{F}_{2^n}$ is defined as

- $a \otimes b = a \cdot g \pmod{f}$.
- Inversion is performed by a variant of the Euclidean algorithm for polynomials.

Characteristic Two Fields

Often write

$$\mathbb{F}_{2^n} = \mathbb{F}_2[x]/f$$

to denote working modulo f.

Set of non-zero elements denoted by $\mathbb{F}_{2^n}^*$

- This is the multiplicative subgroup of the field

Char 2 Example

Let $f = x^6 + x + 1$ (this is irreducible) The finite field of 2^6 elements can then be identified with

- Bit strings of length six bits
- Binary polynomials of degree less than or equal to five

$a = 001101 = x^3 + x^2 + 1$
$b = 101011 = x^5 + x^3 + x + 1$
$a \oplus b = 100110 = x^5 + x^2 + x$

- Since the two x^3 and the two 1 terms cancel, as we are working mod two.
- Notice, we are simply taking the exclusive-or of the bit string representation.

Char 2 Example

Recap $f = x^6 + x + 1$, $a = 001101 = x^3 + x^2 + 1$, $b = 101011 = x^5 + x^3 + x + 1$.

Since f is sparse reduction mod f done using rewriting, as $x^6 = x + 1 \pmod{f}$,

$$\begin{aligned} a \otimes b &= (x^3 + x^2 + 1) \cdot (x^5 + x^3 + x + 1) \\ &= x^8 + x^7 + x^6 + x^4 + x^3 + x^2 + x + 1 \\ &= x^6 \cdot (x^2 + x + 1) + x^4 + x^3 + x^2 + x + 1 \\ &= (x + 1) \cdot (x^2 + x + 1) + x^4 + x^3 + x^2 + x + 1 \\ &= (x^3 + 1) + (x^4 + x^3 + x^2 + x + 1) \\ &= x^4 + x^2 + x. \end{aligned}$$

i.e. $a \otimes b = 010110 = x^4 + x^2 + x$.

Char 2 Example

Since f is assumed irreducible, every polynomial $a \neq 0$ is coprime to f.

Hence, using a binary polynomial version of the extended GCD algorithm we can find u and v so that

$$u \cdot a + v \cdot f = 1 \pmod 2.$$

In which case $a^{-1} = u$ in $\mathbb{F}_{2^n}$.

If $a = x^3 + x^2 + 1$ and $f = x^6 + x + 1$ then taking $u = x^5 + x^3$ and $v = x^2 + x + 1$ gives us

- $u \cdot a + v \cdot f = 1 \pmod 2$

and so

- $a^{-1} = u = x^5 + x^3 = 101000.$

Choice of Defining Polynomial

All char 2 fields of the same degree n are isomorphic.

- This means it does not depend on which polynomial f we take.
- Different f's give different representations of the same thing.

Let $f(x)$ and $g(y)$ be irreducible polynomials of degree n. Then there are polynomial's $r(x)$ and $s(y)$ such that one can map one field into the other via

- $x \pmod{f(x)} \longrightarrow s(y) \pmod{g(y)}$
- $y \pmod{g(y)} \longrightarrow r(x) \pmod{f(x)}$

This means we can select the best irreducible polynomial f for our own implementation.

- Requires the mapping $s(y)$ only when talking to someone elses implementation which uses $g(y)$ instead.

Characteristic Two : Composite Extension

These are fields of the form $\mathbb{F}_{2^{n \cdot m}}$

For a while some people proposed these (of course backed up by patents).

- The IETF standards include such finite fields.
- They provide a number of performance advantages

Problem is that such fields when used in ECC are suspectible to Weil Descent attacks.

- Whilst such attacks are often not practical they cast sufficient concern to mean we no longer use such fields.

Characteristic Two : Prime Extension

$K = \mathbb{F}_{2^p}$ With p prime, eg $p = 163, 191$.

- Trinomial Bases
- Pentanomial Bases
- Normal Bases

All are very good in hardware, normal bases are very good.

All three occur in standards documents ANSI/NIST etc.

- These days Normal Bases less used.

Large Prime Characteristic Fields

Large Prime Characteristic

$K = \mathbb{F}_p$

Can be implemented in a number of ways.

- Montgomery arithmetic (general prime)
- Barrett Reduction (general prime)
- Generalised Mersenne Primes (special prime)

Most popular method for a general modulus is Montgomery arithmetic.

In deployed classical ECC systems the most popular choice are Generalised Mersenne Primes.

GM Primes

GM Primes

$K = \mathbb{F}_p$

Standards bodies have settled on GM-Primes as the main recommend fields.

- GM-Primes give significant performance advantages.
- Their special form means one has significant performance improvements.
- Montgomery Mult takes about $2n(n+1)$ word multiplications, whereas for GM-Primes this is only n^2.

A GM-prime is one of the form

$$p = f(2^{32}) \text{ or } f(2^{64}),$$

where f is a "low weight" polynomial.

- Eg. $p = 2^{192} - 2^{64} - 1$ is popular.

GM Primes

Suppose we take $p = 2^{192} - 2^{64} - 1$ as an example and we want to compute

$$z = x \cdot y \pmod{p}.$$

To perform modular multiplication we first do a standard school book (or Karatsuba) multiplication

$$a = x \cdot y = a_1 2^{192} + a_0$$

where $a_0, a_1 < 2^{192}$.

- This requires at most 36 32-bit word multiplications plus some additions

GM Primes

We now need to produce $z = a \pmod{p}$, but note that mod p we have

$$2^{192} = 2^{64} + 1$$

and so

$$\begin{aligned} a &\equiv a_1(2^{64}+1) + a_0 \pmod{p} \\ &= b_1 2^{192} + b_0, \end{aligned}$$

where $b_0 < 2^{192}$ but $b_1 < 2^{65}$.

We then repeat to obtain

$$\begin{aligned} b &\equiv b_1(2^{64}+1) + b_0 \pmod{p} \\ &= z. \end{aligned}$$

GM Primes

Notice the reduction stage just involved some shifting and addition

- i.e. very cheap operations.

This total time is dominated by the 36 32-bit word multiplications.

If we did Montgomery arithmetic on similar size numbers we would require

$$2 \cdot 6 \cdot (6 + 1) = 84$$

32-bit word multiplications.

Hence the GM-Prime version will be around twice as fast.

OEF Fields

OEF Fields

$K = \mathbb{F}_{p^n}$
These have appeared in a number of papers for classical ECC.
OEF Fields.

- OEF fields choose p close to the word size.
- The equation defining K over $\mathbb{F}_p$ is chosen to be very simple,

$$x^n - 2.$$

OEF Fields give very good implementations.

- Very fast field inversion.

Mainly utilized in pairing based systems, for the second field

Curve Arithmetic

Curve Arithmetic

Points can be added/doubled in a number of formats:

Affine:

- $P = (X, Y)$.

Projective (Standard):

- $P = (X, Y) = (x/z, y/z)$.

Projective (Jacobian):

- $P = (X, Y) = (x/z^2, y/z^3) = (x, y, z)$.

Chudnovsky:

- As Jacobian but store $P = (x, y, z, z^2, z^3)$.

Lopez-Dahab:

- $P = (X, Y) = (x/z, y/z^2) = (x, y, z)$

Mixed:

- A combination of any of the above.

Curve Arithmetic

Standard projective coordinates are rarely used.

We will concentrate on

- Affine,
- Jacobean Projective/Lopez-Dahab
- A mixture of the two.

Main problem with Affine is that we require field inversions.

- Field inversions are generally much slower than multiplications.

Projective coordinates allow us to trade a number of multiplications for an inversion.

We shall see later that its is point doubling which is most important.

Cost of Curve Arithmetic: Char p

In odd characteristic it is often best to use Jacobean Projective coordinates:

Operation	Affine	Projective	Mixed
Addition	3M + 1I	16M	11M
Doubling	4M + 1I	10M	n/a

In next few slides we give the formulae for computing

- $P_3 = P_1 + P_2$

with

- $P_i = (X_i, Y_i, Z_i)$

when the curve is given by

$$Y^2 = X^3 + AX + B.$$

Projective Addition Formulae: Char p

$$\begin{array}{lcll}
\lambda_1 & = & X_1 Z_2^2 & 2M \\
\lambda_2 & = & X_2 Z_1^2 & 2M \\
\lambda_3 & = & \lambda_1 - \lambda_2 & \\
\lambda_4 & = & Y_1 Z_2^3 & 2M \\
\lambda_5 & = & Y_2 Z_1^3 & 2M \\
\lambda_6 & = & \lambda_4 - \lambda_5 & \\
\lambda_7 & = & \lambda_1 + \lambda_2 & \\
\lambda_8 & = & \lambda_4 + \lambda_5 & \\
Z_3 & = & Z_1 Z_2 \lambda_3 & 2M \\
X_3 & = & \lambda_6^2 - \lambda_7 \lambda_3^2 & 3M \\
\lambda_9 & = & \lambda_7 \lambda_3^2 - 2X_3 & \\
Y_3 & = & (\lambda_9 \lambda_6 - \lambda_8 \lambda_3^3)/2 & 3M \\
\hline
 & & & 16M
\end{array}$$

Projective Addition Formulae: Char p

In previous slide

$\lambda_3 = 0$ if and only if $P_1 = \pm P_2$

$\lambda_3 = \lambda_6 = 0$ if and only if $P_1 = P_2$

- In this case need to execute a point doubling (see later)

A mixed addition is when $Z_1 = 1$ in which case we simplify the formulae as in the next slide

Projective (Mixed) Addition Formulae: Char p

$$\begin{array}{rcll}
\lambda_1 & = & X_1 Z_2^2 & 2M \\
\lambda_3 & = & \lambda_1 - X_2 & \\
\lambda_4 & = & Y_1 Z_2^3 & 2M \\
\lambda_6 & = & \lambda_4 - Y_2 & \\
\lambda_7 & = & \lambda_1 + X_2 & \\
\lambda_8 & = & \lambda_4 + Y_2 & \\
Z_3 & = & Z_2 \lambda_3 & 1M \\
X_3 & = & \lambda_6^2 - \lambda_7 \lambda_3^2 & 3M \\
\lambda_9 & = & \lambda_7 \lambda_3^2 - 2X_3 & \\
Y_3 & = & (\lambda_9 \lambda_6 - \lambda_8 \lambda_3^3)/2 & 3M \\
\hline
 & & & 11M
\end{array}$$

Projective Doubling Addition Formulae: Char p

A point doubling is performed via

$$\begin{array}{rcll}
\lambda_1 & = & 3X_1^2 + AZ_1^4 & 4M \\
Z_3 & = & 2Y_1Z_1 & 1M \\
\lambda_2 & = & 4X_1Y_1^2 & 2M \\
X_3 & = & \lambda_1^2 - 2\lambda_2 & 1M \\
\lambda_3 & = & 8Y_1^4 & 1M \\
Y_3 & = & \lambda_1(\lambda_2 - X_3) - \lambda_3 & 1M \\
\hline
 & & & 10M
\end{array}$$

Projective Doubling Addition Formulae: Char p

Recall we said usually $A = -3$ then

$$\begin{array}{rcll}
\lambda_1 & = & 3X_1^2 + AZ_1^4 = 3(X_1 + Z_1^2)(X_1 - Z_1^2) & 2M \\
Z_3 & = & 2Y_1Z_1 & 1M \\
\lambda_2 & = & 4X_1Y_1^2 & 2M \\
X_3 & = & \lambda_1^2 - 2\lambda_2 & 1M \\
\lambda_3 & = & 8Y_1^4 & 1M \\
Y_3 & = & \lambda_1(\lambda_2 - X_3) - \lambda_3 & 1M \\
\hline
 & & & 8M
\end{array}$$

Cost of Curve Arithmetic: Char 2

In even characteristic it is often best to use Lopez-Dahab coordinates:

Operation	Affine	Lopez-Daheb	Mixed
Addition	2M + 1I	13 M	8 M
Doubling	2M + 1I	4 M	n/a

Note squaring is free in even characteristic.

In next few slides we give the formulae for computing

- $P_3 = P_1 + P_2$

with

- $P_i = (X_i, Y_i, Z_i)$

when the curve is given by

$$Y^2 + XY = X^3 + AX^2 + B.$$

Projective Addition Formulae: Char 2

$$
\begin{array}{lcll}
\lambda_1 & = & X_1 Z_2 & 1M \\
\lambda_2 & = & X_2 Z_1 & 1M \\
\lambda_3 & = & \lambda_1 + \lambda_2 & \\
\lambda_4 & = & \lambda_1^2 & \text{free} \\
\lambda_5 & = & \lambda_2^2 & \text{free} \\
\lambda_6 & = & \lambda_4 + \lambda_5 & \\
\lambda_7 & = & Y_1 Z_2^2 & 1M \\
\lambda_8 & = & Y_2 Z_1^2 & 1M \\
\lambda_9 & = & \lambda_7 + \lambda_8 & \\
\lambda_{10} & = & \lambda_3 \lambda_9 & 1M \\
Z_3 & = & Z_1 Z_2 \lambda_6 & 2M \\
X_3 & = & \lambda_1(\lambda_8 + \lambda_5) + \lambda_2(\lambda_7 + \lambda_4) & 2M \\
Y_3 & = & (\lambda_1 \lambda_{10} + \lambda_7 \lambda_6)\lambda_6 + (\lambda_{10} + Z_3) X_3 & 4M \\
\hline
 & & & 13M
\end{array}
$$

Projective (Mixed) Addition Formulae: Char 2

$$\begin{array}{lcll}
\lambda_1 & = & Z_2 Y_1 + Y_2 & 1M \\
\lambda_2 & = & Z_2 X_1 + X_2 & 1M \\
\lambda_3 & = & Z_2 \lambda_2 & 1M \\
Z_3 & = & \lambda_3^2 & \text{free} \\
\lambda_5 & = & Z_3 X_1 & 1M \\
\lambda_6 & = & X_1 + Y_1 & \\
X_3 & = & \lambda_1^2 + \lambda_3(\lambda_1 + \lambda_2^2 + A\lambda_3) & 2M \\
Y_3 & = & (\lambda_5 + X_3)(\lambda_3\lambda_1 + Z_3) + \lambda_6 Z_3^2 & 3M \\
\hline
 & & & 9M
\end{array}$$

If $A = 1$ then this reduces to $8M$.

Projective Doubling Addition Formulae: Char 2

A point doubling is performed via

$$\begin{array}{rcll}
\lambda_1 & = & X_1^2 & \text{free} \\
\lambda_2 & = & \lambda_1 + Y_1 & \\
\lambda_3 & = & X_1 Z_1 & 1M \\
Z_3 & = & \lambda_3^2 & \text{free} \\
\lambda_5 & = & \lambda_2 Z_3 & 1M \\
X_3 & = & \lambda_2^2 + \lambda_3 + A Z_3 & 1M \\
Y_3 & = & (Z_3 + \lambda_5) X_3 + \lambda_1^2 Z_3 & 2M \\
\hline
 & & & 5M
\end{array}$$

Which reduces to $4M$ if we choose $A = 1$.

Projective Formulae Summary

Odd Characteristic

Operation	Affine	Projective	Mixed
Addition	3M + 1I	16M	11M
Doubling	4M + 1I	10M	n/a

Even Characteristic

Operation	Affine	Lopez-Daheb	Mixed
Addition	2M + 1I	13 M	8 M
Doubling	2M + 1I	4 M	n/a

Doubling is very fast in even characteristic.

- This can make up for the slow software field arithmetic in a final implementation.

Point Multiplication

Point Multiplication

The basic cryptographic operation is to compute

$$Q = [d]P$$

for some integer d and point P.

The binary method:

- $Q = 0$.
- For $j = \log_2(d) - 1$ to 0
    - $Q = [2]Q$.
    - If $d_j = 1$ then $Q = Q + P$.

This requires

- Exactly $\log_2(d)$ point doublings.
- About $\log_2(d)/2$ general point additions.
    - If P affine and Q projective used mixed addition

We can reduce the number of general point additions.

Point Multiplication

m-ary method

The binary method uses a fixed window of size 2.
The *m*-ary method uses a fixed window of size $m = 2^r$.

- Taking *m* bits at a time.

Requires precomputation of

$$2^{r-1} - 1$$

general point additions.

Then requires

- $\log_2(d)$ doublings.
- About $\log_2(d)/r$ general point additions.

Point Multiplication

Sliding Window Method

This method slides the window of length m across runs of zero's in the binary expansion.

Requires precomputation of

$$2^{r-1} - 1$$

general point additions.

Then requires

- $\log_2(d)$ doublings.
- About $\log_2(d)/(r+1)$ general point additions.

Point Multiplication

Signed Window Methods

For elliptic curves negation comes for free

$$-P = (x, -y) \text{ or } (x, y + x).$$

Hence we could used a signed binary representation.

$$7 = 2^2 + 2 + 1 \text{ or } 7 = 2^3 - 1.$$

This allows us to reduce the number of point additions even further

- In both the *m*-ary and the sliding window algorithms.

Signed Sliding Window Method

The following precomputation is done once for each point P;

Precomputation

- $P_1 = P$, $P_2 = [2]P$.
- For $i = 1$ to $2^{r-2} - 1$
    - $P_{2i+1} = P_{2i-1} + P_2$..
- $Q = P_{d_{l-1}}$.

These are computed in affine coordinates

Next we encode the number d Set

$$d = \sum_{i=0}^{l-1} d_i 2^{e_i}$$

with $e_{i+1} - e_i \geq r$ and

$$d_i \in \{\pm 1, \pm 3, \ldots, \pm 2^{r-1} - 1\}.$$

Signed Sliding Window Method

Main Loop

```
For $i = l - 2$ to 0
    $Q = [2^{e_{i+1} - e_i}]Q$.
    If $l_i > 0$ then $Q = Q + P_{d_i}$.
    Else $Q = Q - P_{-d_i}$.
$Q = [2^{e_0}]Q$.
```

The Q is held in projective coordinates

- Since P_i are affine can use mixed addition
- Can use efficient projective doubling formulae

Point Multiplication

The signed sliding window method generally comes out to be the fastest.

Remember
In all cases CPU time is dominated by the time to compute

$$\log_2(d)$$

point doublings.

Lesson
Optimise the doubling operation at all times.

- This is often ignored.

Point Multiplication

Notice that doubling in char 2 requires less multiplications than in char p.

This can often lead (depending on the processor or the implementation) that the relative advantage of char p field multiplication can be cancelled out by the doubling operation.

This will become even more pronounced as the new instruction set extensions to Intel and other chips become more prevalent, since these will provide native carry-free multipliers.

- i.e. char 2 field multiplications will run as fast as char p field multiplications
- Indeed possibly faster.

Key Agreement

Elliptic Curve Diffie-Hellman

Easiest to understand of all the protocols.

Two people, Alice and Bob, want to agree a shared secret.

$E(\mathbb{F}_q)$ is an elliptic curve over a finite field for which ECDLP is hard.

P is a point of large prime order.

EC-DH

$$\begin{array}{ccc} \text{Alice} & & \text{Bob} \\ x & \overset{[x]P}{\Longrightarrow} & [x]P \\ [y]P & \overset{[y]P}{\Longleftarrow} & y \end{array}$$

Alice can now compute

$$K_A = [x]([y]P) = [xy]P$$

Bob can now compute

$$K_B = [y]([x]P) = [xy]P$$

and

$$K_A = K_B$$

Very small bandwidth if one uses point compression.

EC-DHP

Given

$$[x]P \text{ and } [y]P$$

the problem of recovering

$$[xy]P$$

is called the Elliptic Curve Diffie-Hellman Problem (ECDHP).

If we can solve ECDLP then we can solve ECDHP.
It is unknown if the other implication holds.

A proof of equivalence of the DHP and DLP for many black box groups follows from work of Boneh, Maurer and Wolf.
This proof uses elliptic curves in a crucial way.

- EC-DH is standardised in ANSI X9.63.

EC-DH

In practice life is not so simple, for example in EC-DH we have a man-in-the-middle attack

$$\begin{array}{ccccc} \text{Alice} & & \text{Eve} & & \text{Bob} \\ x & \longrightarrow & [x]P & & \\ [m]P & \longleftarrow & m & & \\ [xm]P & & [xm]P & & \\ & & n & \longrightarrow & [n]P \\ & & [y]P & \longleftarrow & y \\ & & [yn]P & & [yn]P \end{array}$$

Alice agrees a key with Eve, thinking it is Bob Bob agrees a key with Eve, thinking it is Alice

Eve can now examine communications as they pass through her (she acts as a router).

Diffie-Hellman

Diffie-Hellman on its own is not enough.

For example how does Alice know who she is agreeing a key with ?

- Is it Bob or Eve ?

One way around is for

- Alice to sign her message to Bob
- Bob to sign his message to Alice.

In that way both parties know who they are talking to.

Signed Diffie-Hellman

Assuming we can construct secure signature schemes we have now solved the key distribution problem:

Authentic public keys are obtained from a CA.

Then secure session keys are obtained using signed Diffie-Hellman

$$\begin{array}{ccc} \text{Alice} & & \text{Bob} \\ ([a]P, \text{Sign}_{\text{Alice}}([a]P)) & \longrightarrow & \\ & \longleftarrow & ([b]P, \text{Sign}_{\text{Bob}}([b]P)) \end{array}$$

However, it is more common to use the STS-protocol in this situation

STS Key Agreement Protocol

When using signed Diffie–Hellman it is common to adopt the Station-to-Station protocol, or STS protocol

- $A \longrightarrow B : [x]P$
- $B \longrightarrow A : [y]P, \{Sig_B([y]P, [x]P)\}_{K_{ab}}$
- $A \longrightarrow B : \{Sig_A([x]P, [y]P)\}_{K_{ab}}$

where $K_{ab} = g^{xy}$

STS provides forward secrecy, but has some subtle problems

Has unknown key share attack

- This attack requires adversary to obtain "invalid" certificates from a CA
- Very damaging if duplicate signatures can be found

MQV Protocol

System setup

Alice and Bob generate a public/private key pair each

$$(A = [a]P, a) \text{ and } (B = [b]P, b).$$

Via some means (eg a certificate)

- Bob knows A is authentic
- Alice knows B is authentic

They now want to agree on a secret session key to which they both contribute a random nonce

- nonce = number which is used once and then thrown away

Use of the nonce's provides them with forward secrecy

MQV Authenticated Key Exchange

This is the most efficient deployed authenticated key exchange protocol

- Most analysed authenticated key exchange mechanism available
- NSA like this one

The key exchange Alice and Bob now generate a public/private ephemeral key pair each

$$(C = [c]P, c) \text{ and } (D = [d]P, d).$$

They exchange C and D.

Hence to some extent this looks like a standard Diffie-Hellman exchange with no signing.

However the final session key will also depend on A and B.

MQV

Determining the Session Key

Assume you are Alice

- You know A, B, C, D,a and c

Let l denote half the bit size of the group G

- e.g. $l = 160/2 = 80$

Shared secret K computed via

- Convert C to an integer i
- Put $s = (i \pmod{2^l}) + 2^l$
- Convert D to an integer j
- Put $t = (j \pmod{2^l}) + 2^l$
- Put $h = c + sa$.
- Put $K = [h]([t](DB)$

Why does MQV this work ?

Note that s and t seen be Alice, are swapped when seen by Bob,

- $s_{Alice} = t_{Bob}$
- $t_{Alice} = s_{Bob}$

Let

- h_{Alice} denote the h seen by Alice
- h_{Bob} denote the h seen by Bob

Then

- $P = g^{h_{Alice} \cdot h_{Bob}}$

You should check this for yourself

MQV however still suffers from a unknown key-share attack

- But a very subtle attack, hence probably not important in practice

Encryption

Hybrid Encryption : KEMs and DEMs

Before introducing ECIES we recap on the modern method for creating public key encryption schemes.

Most public key schemes are used in a hybrid manner.

- A public key system is used to encrypt a symmetric key (key encapsulation mechanism).
- The symmetric key is used to encrypt the actual data (data encapsulation mechanism).

This is now formalised: the KEM-DEM or hybrid encryption methodology.

Key Encapsulation Mechanisms

Key Encapsulation Mechanism

A KEM is an algorithm which takes as input a public key y and outputs a pair (K, C) where

- K is a key for a symmetric encryption function (e.g. DES or AES) and
- C is an encapsulation (encryption) of K using y.

The inverse, decapsulation algorithm takes as input (C, x) where

- C is an encapsulation under y of some key K - or it should be! - and
- x is the private key corresponding to y.

It outputs

- either $\perp$ if C is an invalid encapsulation, or
- K if C is an encapsulation of the key K.

Data Encapsulation Mechanisms

Data Encapsulation Mechanism

A DEM is a symmetric algorithm which on input of

- a message M and
- a symmetric key K

outputs an encryption E of M using key K.

The inverse operation takes as input

- (E, K)

and outputs either

- $\perp$ if E is an invalid ciphertext or
- M if E is an encryption of M under the key K.

A KEM-DEM Hybrid Cipher

Using the primitives we have been discussing, a KEM-DEM hybrid encryption scheme can then be created as follows.

Encryption

- $(K, C) \longleftarrow \mathrm{KEM}(y)$
- $E \longleftarrow \mathrm{DEM}(M, K)$
- Return (C, E)

Decryption

- $K \longleftarrow \mathrm{KEM}^{-1}(C, x)$
- If $K = \perp$ then return $\perp$
- $M \longleftarrow \mathrm{DEM}^{-1}(E, K)$
- If $M = \perp$ then return $\perp$
- Return M

DEM : Construction

To construct a secure DEM we take

- a secure (under passive attack) block cipher E and
- a secure (cannot produce a MAC without the corresponding key) MAC function MAC.

The function $\mathrm{DEM}(M, K)$ is then constructed as follows.

- Split K into k_0 and k_1.
- $c_0 \longleftarrow E(M, k_0)$.
- $c_1 \longleftarrow \mathrm{MAC}(c_0, k_1)$.
- Return $C = (c_0, c_1)$.

KEM-DEM Security Properties

It can be shown that if

- a KEM is secure

and

- a DEM is secure

then the combined KEM-DEM hybrid scheme is IND-CCA2 secure.

Thus the KEM-DEM approach to public key encryption allows us to build schemes in a modular fashion:

- design a secure KEM;
- design a secure DEM;
- combine them to obtain a secure encryption scheme.

Each component can be designed independently.

EC-IES

This is the standard ECC encryption algorithm.

- Based on Abdalla, Bellare and Rogaway's DHAES protocol
- Secure under a non-standard assumption (Abdalla et. al.)
- Secure under in the ROM (Abdalla et. al.)
- Secure under in the Generic Group Model (Smart)

IES stands for integrated encryption scheme

- The scheme works like static Diffie-Hellman followed by symmetric encryption.
- We use Diffie-Hellman as a KEM
- Then encrypt the message with a DEM

EC-IES Components

A symmetric encryption scheme $\mathrm{SYM} = (E_k, D_k)$,

- Key space K_1

A MAC function MAC_k

- Key space K_2.

A key derivation function V.

- The key derivation function V will map group elements
- to the key space of both the encryption and MAC functions.

The scheme ECIES is defined as a triple of randomised algorithms,

- $\{$**keygen, enc, dec**$\}$.

EC-IES KeyGen

- $d \leftarrow \{1, \ldots, q\}$.
- $Q \leftarrow [d]P$.
- Return (Q, d).

EC-IES Encryption

- $k \leftarrow \{1, \ldots, q\}$.
- $U \leftarrow [k]P$.
- $T \leftarrow [k]Q$.
- $(k_1, k_2) \leftarrow V(T)$
- $c \leftarrow E_{k_1}(m)$
- $r \leftarrow \mathrm{MAC}_{k_2}(c)$
- Return $e \leftarrow U \| c \| r$.

The cipher text is (U, c, r).

- U is needed to agree a key
- c is the actual encrypted message
- r is used to avoid adaptive chosen ciphertext attacks

The data item U can be compressed to reduce bandwidth.

EC-IES Decryption

- Parse e as $U \| c \| r$
- $T \leftarrow [d]U$
- $(k_1, k_2) \leftarrow V(T)$
- If $r \neq \mathrm{MAC}_{k_2}(c)$
    - Return **Invalid**
- $m \leftarrow D_{k_1}(c)$.
- Return m.

ECIES

EC-IES makes it easy to encrypt long messages

Standardized in a number of places

- ANSI X9.63
- IEEE P1363
- SEC 1
- etc

Signatures

EC-DSA

Variant of the American DSA algorithm as specified in NIST FIPS 186.

FIPS 186.2 contains the new version including EC-DSA,

- This is also in ANSI X9.62, IEEE P1363 and SECG

With all digital signature algorithms one actually signs a hash of the message, M.

- For ECC this hash function is always $SHA-1/SHA-2$.
- $SHA-1$ (resp. $SHA-2$) takes an arbitrary length input and produces a 160 (resp 256 etc) bit output.
- We interpret this output bit string as a number.

We also interpret the x-coordinate of a point as a number.

- Even when $K = \mathbb{F}_{2^p}$.

EC-DSA : System set up

E an elliptic curve over $K = \mathbb{F}_{p^n}$.
P a point of large prime order, q.

$$\#E(K) = hq$$

h is called the cofactor.

The set $\{K, E, q, h, P\}$ is called the domain parameters.

Private Key : $d \in_R [1, \ldots, q-1]$. Public Key : $Q = [d]P$.

EC-DSA : Signing

Choose $k \in_R [1, \ldots, q-1]$.
Compute $[k]P = (x, y)$.
Convert x to an integer, r, mod q.
If $r \equiv 0$ then goto beginning.
Put $e = SHA-1(M)$.
Compute

$$s \equiv (e + dr)/k \pmod{q}.$$

If $s \equiv 0$ then goto beginning.
Return (r, s) as the signature.

EC-DSA : Verifying

Put $e = SHA - 1(M)$.
Reject if $r, s \notin [1, \ldots, q-1]$.
Compute

$$\begin{aligned} u_1 &\equiv e/s \pmod{q}. \\ u_2 &\equiv r/s \pmod{q}. \end{aligned}$$

Set $R = (x, y) = [u_1]P + [u_2]Q$.
Reject if R is at infinity.
Convert x to an integer, l, modulo q.
Accept if and only if $r \equiv l$.

EC-DSA

Need to make sure ephemeral exponent is truly random.

Signing is much faster than RSA

- Verification is slower.

Size of signature is much smaller than RSA.

Scales much better than RSA or DSA over time.

Schnorr Signatures

An important DLP based signature scheme is that of Schnorr.

- It occurs in many ZK proofs and as parts of other protocols.
- It is the simplest among the DLP based schemes that are provably secure.
- The signing and verifying operations are simpler than those for DSA.
- Is the basis for many other protocols.
- Standardisation by ISO

Each user generates a secret signing key x at random and such that

- $0 < x < q$.

Public key is $Q = xP$.

Schnorr Signatures : Signing

To sign a message M the signer proceeds as follows.

- Signer chooses a random ephemeral key: $0 < k < q$.
- Signer computes $R = kP$.
- Signer computes one-way hash $m = H(R||M)$.
- Finally, signer computes

$$s = (k + mx) \pmod{q}.$$

The signature on M is the pair (m, s).

Schnorr Signatures : Verification

To verify a signature (m, s) on a message M under public key Q, the verifier proceeds as follows.

The verifier computes

$$R' = sP - mQ.$$

If the signature is valid we have

$$R' = (k + mx)P - xmP = kP.$$

So, the verifier accepts signature if and only if

$$m = H(R' \| M).$$

ECDLP

How hard is the ECDLP?

Black box group means there is no information about the representation.

- Only generic algorithms are available.

Theorem: (V. Shoup)
In a black box group of prime order L it takes at least $O(\sqrt{L})$ operations to solve the discrete logarithm problem.

Compare to factoring where underlying problem is sub-exponential

Generic Algorithms

Baby step giant step (Shanks)

Want to find $0 \leq \lambda < L$ such that $Q = \lambda P$ in $E(\mathbb{F}_q)$.

Put $M = \lceil \sqrt{L} \rceil$ (or $M = \lceil \sqrt{L/2} \rceil$).

Make a list of baby steps:

- $\mathcal{O}_E, P, 2P, \ldots, MP$.

Take giant steps $Q - MP, Q - 2MP, \ldots$ until find a match

$$Q - \lambda_1 MP = \lambda_0 P$$

with the list.

Then $\lambda = \lambda_0 + M\lambda_1$.

- Time: $O(\sqrt{L})$.
- Memory: $O(\sqrt{L})$.

Pollard methods

Use deterministic random walks in $E(\mathbb{F}_q)$:

- Partition $E(\mathbb{F}_q)$ into 2^n sets $G_1, \ldots, G_{2^n}$.
- Construct 2^n random points $P_i = \alpha_i P$.
- Random walk $X \mapsto (X + P_i \text{ if } X \in G_i)$.

Method:

- Start at $X = P$ and take $O(\sqrt{L})$ steps in random walk and store the final value $Y = \alpha P$.
- Start at $X = Q$ and take steps in walk until hit Y.
- Have $Q + \alpha' P = \alpha P$.

- Time: $O(\sqrt{L})$.
- Memory: $O(1)$.

Parallel Pollard (Van Oorschot-Wiener)

Distinguished point set $\mathcal{D}$, size $\theta \# E(\mathbb{F}_q)$.

Method: Suppose we have M processors in parallel.

- Each processor starts at a random point $X = \alpha P + \beta Q$ and walks in the group.
- Every time a distinguished point X is encountered then send (X, α, β) to the central server.
- When the server receives (X, α, β) and (X, α', β') then can solve for discrete logarithm.

- Time: $\sqrt{\pi L/2}/M + L/(\theta \# E(\mathbb{F}_q))$.
- Server memory: $\theta\sqrt{L}$.

In practice: $L \sim 2^{100}$, $M \sim 2^{10}$ and $\theta = 2^{-30}$.

Equivalence classes (W-Z/G-L-V)

Example :

- For $P = (x, y)$ have $-P = (x, -y)$.
- Impose a canonical choice of representative for elements of the set $E(\mathbb{F}_q)/\langle \pm 1 \rangle$
- Define the random walk on this set instead.
- Pollard methods are faster by a factor of $\sqrt{2}$.

Example:

- Consider a subfield curve $E/\mathbb{F}_2$ with discrete logarithm problem in $E(\mathbb{F}_{2^l})$ (Koblitz curve)
- Action of $\pm Frob_2$ gives equivalence classes of size $2l$.
- So method faster by factor $\sqrt{2l}$.

Koblitz curves are recommended in some standards eg ANSI, NIST etc

Special Algorithms

Special Attacks

There are a number of special attacks.

- Only apply to certain curves
- Analogous to weak RSA keys

Unlike weak RSA keys, weak ECC keys are easily detected by any user

- i.e. can be detected by anyone and not just the person who makes the key.

Menezes-Okamoto-Vanstone/Frey-Rück

Construct group homomorphism

$$E(\mathbb{F}_q) \longrightarrow \mathbb{F}_{q^k}^*$$

where k is the smallest integer such that the exponent of $E(\mathbb{F}_q)$ divides $q^k - 1$.

Can solve discrete logarithm problem in $\mathbb{F}_{q^k}^*$ using an index calculus algorithm of subexponential complexity (in q^k).

E supersingular implies $k \leq 6$.

General case $k \sim q$.

- But are some special cases for ordinary curves.

Menezes-Okamoto-Vanstone/Frey-Rück

There is a pairing, the (modified) Tate pairing, such that for supersingular curves

$$t : \begin{cases} E(\mathbb{F}_q) \times E(\mathbb{F}_q) & \longrightarrow & \mathbb{F}_{q^k}^* \\ (P, Q) & \longmapsto & f_{n,P}(Q)^{(q^k-1)/n} \end{cases}$$

where

- $(f_{n,P}) = n(P) - n(\mathcal{O})$
- $n = \#E(\mathbb{F}_q)$
- $t(P, Q)$ is bilinear
- If $P, Q \neq \mathcal{O}$ then $t(P, Q) \neq 1$

Menezes-Okamoto-Vanstone/Frey-Rück

To solve

$$Q = \lambda P$$

Compute

- $g = t(P, P)$
- $h = t(Q, P)$

Try to solve, in the finite field,

$$\begin{aligned} h &= t(Q, P) \\ &= t(\lambda P, P), \\ &= t(P, P)^{\lambda}, \\ &= g^{\lambda}. \end{aligned}$$

Menezes-Okamoto-Vanstone/Frey-Rück

For general curves the modified Tate pairing is defined as

$$t : \begin{cases} E(\mathbb{F}_q) \times \overline{E}(\mathbb{F}_{q^e}) & \longrightarrow & \mathbb{F}_{q^k}^* \\ (P, Q) & \longmapsto & f_{n,P}(Q)^{(q^k-1)/n} \end{cases}$$

where

- $(f_{n,P}) = n(P) - n(\mathcal{O})$
- $n = \#E(\mathbb{F}_q)$
- $t(P, Q)$ is bilinear
- If $P, Q \neq \mathcal{O}$ then $t(P, Q) \neq 1$
- $e = k/d$ where d is the largest possible twist
- $\overline{E}$ is a d-th twist of E, which is a curve defined over $\mathbb{F}_{q^e}$.

Semaev/Smart/Araki-Satoh

Suppose $E(\mathbb{F}_p)$ has exactly p points.

Construct a group homomorphism

$$E(\mathbb{F}_p) \longrightarrow \mathbb{F}_p^+.$$

Methods:

- Using p-adic logarithm and p-adic lift.
- Take a function f such that $(f) = p(P) - p(\mathcal{O})$ and consider the holomorphic differential $\omega = \frac{1}{f} df$.

Discrete logarithm problem in $\mathbb{F}_p^+$ solved using Euclid's algorithm.

Weil Descent

Only (currently) applies to fields of characteristic two.

If curve defined over $\mathbb{F}_{q^n}$ for a small value of n can reduce ECDLP to a HCDLP.

For some values of n, eg $n = 4$ this weakens the curve.

- Work of Frey, Galbraith, Gaudry, Hess and Smart

Values of n of $4, 5, 6$ sometimes chosen for efficiency reasons.

Note, only standard which uses such curves is IPSec (I think)

New Techniques

The most successful of the more modern techniques (post 2005) have been those in the Semaev/Gaudry/Diem family.

- Use a combination of index calculus, division polynomials and Groebner basis.

Almost all are non-practical but they obtain sub-exponential complexity for infinite familties of curves over fields of the form

$$\mathbb{F}_{p^n}$$

where p and n lie in certain regions.

- Sort of "medium characteristic" fields.

Gaudry's Method

For curves over $\mathbb{F}_{q^n}$

For fixed n, but with q tending to infinity, Gaudry obtains a complexity of

$$O(q^{2-2/n}).$$

Comparing to Pollard rho of $O(q^{n/2})$ we see that for $n = 4$ this is more efficient.

- But is totally impractical

Diem's Method

If $a > 2 + \epsilon$ and $(2 + \epsilon)n^2 \leq \log_2(q) \leq an^2$ then obtain

$$\exp(O(1) \cdot (\log(q^n))^{2/3})$$

i.e. $L_{q^n}(1/3, c \cdot \sqrt{a})$ for some constant c.

This generalises Gaudry's result.

We essentially obtain a polynomial algorithm in q as long as $\log_2(q)$ is larger than $2n^2$.

- Hence if q is subexponential in q^n then we get a subexponential algorithm.

Again the method is totally impractical, even for small values of n and q.

How big?

To compare ECC against other technologies we use the following table provided by NIST

Block Cipher Key Size	Example Block Cipher	ECC Key Size	RSA Key Size
80	SKIPJACK	163	1024
128	AES (small)	283	3072
192	AES (medium)	409	7680
256	AES (large)	571	15360

ECC at 571 bits is usable, RSA at 15360 bits is not.

Although 571 is really huge, conservative managers may want to go for the highest level of security possible.

Counting Points

Counting points

To use an elliptic curve in a real system we first need to know

$$\#E(\mathbb{F}_q).$$

For some curves this is easy

- Koblitz curves

For others we need to be more clever to compute this number

Frobenius endomorphism

$$\pi : (x, y) \mapsto (x^q, y^q).$$

Characteristic polynomial

$$\pi^2 - t\pi + q = 0.$$

Then have $\#E(\mathbb{F}_q) = q + 1 - t$.
Hasse: $|t| \leq 2\sqrt{q}$.
Computing $\#E(\mathbb{F}_q)$ is equivalent to computing t.

Schoof

Idea:
Compute the value of t modulo small primes (or prime powers) l
Recover t using the Chinese remainder theorem and the bound $|t| \leq 2\sqrt{q}$.

Very complicated algorithm.

- Can be made to be very efficient using ideas of Elkies and Atkin
- Can compute $\#E(\mathbb{F}_q)$ for a given curve in a matter of seconds for most interesting values of q.

Satoh

Satoh in 1998 invented a new method which is better than Schoof for fields $\mathbb{F}_{p^n}$ of small characteristic p, eg characteristic two,

Lifts the curve to a p-adic extension.

Applies a p-isogeny n times, to obtain an isogeny cycle.

Use this to write down t modulo p^n.

Note : A 2-isogeny is given by the Arithmetic-Geometric mean as any standard textbook on elliptic integrals (or computing π) will tell you.

- Thus Satoh's algorithm gives rise to the AGM method of Harley-Gaudry from 2001
- AGM method only useful for characteristic two.

Any Questions?