Elliptic Curve Cryptography with Derive

Johann Wiesenbauer
Vienna University of Technology

DES-TIME-2006, Dresden
General remarks on Elliptic curves

- Elliptic curves can be described as nonsingular algebraic curves of degree 3 over some field $\mathbb{F}$. In particular, they come in the hierarchy of algebraic curves right after lines and conic sections, which are of degree 1 and 2, respectively.

- Topologically, elliptic curves are of genus 1, again higher by only 1 than lines and conic sections, which are of genus 0.
Algebraically, elliptic curves belong to the so-called abelian varieties (of dimension 1). This means that it is possible to define an addition on their sets of points, which turns them into abelian groups. Those groups play an important role in many applications, in particular in elliptic curve cryptography (ECC).

Last but not least, elliptic curves are also very useful in dealing with certain Diophantine equations (FLT, congruent numbers, etc.)
Definition of Elliptic Curves

- In general, elliptic curves are given by a so-called Weierstrass equation, which is of the following general form
  \[ F(x, y) = y^2 + a_1xy + a_3y - x^3 + a_2x^2 + a_4x + a_6 = 0 \]
  where the coefficients \( a_i \) are in some field \( F \), along with the point \( O \) at infinity. Nonsingularity means here that the system \( F(x, y) = F_x(x, y) = F_y(x, y) = 0 \) has no solutions in any extension field of \( F \).

- If \( \text{char}(F) \neq 2, 3 \) one can even assume the so-called short Weierstrass-Form
  \[ y^2 = x^3 + ax + b \]
  which will be used exclusively in the following. Nonsingularity means here that the discriminant \( 4a^3 + 27b^2 \) of the polynomial on the right has no multiple roots in any extension field of \( F \).
Just a word about the important case char(F)=2.

- In the case char(F)=2, we have two types of elliptic curves:
  \[ y^2 + cy = x^3 + ax + b \]  and  \[ y^2 + xy = x^3 + ax + b. \]
- The first type leads to the supersingular curves, which are not suitable for cryptographic purposes due to the so-called MOV-attack (MOV=Menezes-Okamato-Vanstone), which can solve the DLP in subexponential time.
- As for the second type, we have again a splitting into two cases, as one can consider standard polynomial bases \( 1, \alpha, \alpha^2, \ldots, \alpha^{m-1} \) or so-called normal bases \( \alpha, \alpha^2, \alpha^4, \ldots, \alpha^{q/2} \), where \( q=2^m \) is the number of elements of the field \( F \) and \( \alpha \in F \). (The main advantage of normal bases is that squaring is only a cyclic rotation!)
Elliptic curves over $\mathbb{R}$
(splitting or not splitting)

$y^2 = x^3 - 5x + 2$

$y^2 = x^3 - x + 2$
The curves below of degree 3 are not elliptic curves due to singularities (cusps and double points)
1. If $P=O$ or $Q=O$, then $P+Q=Q$ or $P+Q=P$, respectively. In other words $O$ is a neutral element for $+$. 

2. If $P$ and $Q$ are mirror images w.r.t. the $x$-axis, then set $P+Q = O$. Hence, in this case $P$ and $Q$ are inverse to each other.

3. If neither 1 or 2 is true, then $R=P+Q$ is defined as can be seen from the drawing, i.e. as the mirror point w.r.t. $x$-axis of the intersection of the chord (or tangent, if $P=Q$) with the curve.
A closer look at case 3

Let $P=(x_1,y_1)$ and $Q=(x_2,y_2)$, the the slope $k$ of the secant or tangent through $P$ and $Q$ is given by the formula

$$k = \begin{cases} 
\frac{y_2 - y_1}{x_2 - x_1}, & \text{if } x_1 \neq x_2 \\
\frac{3x_1 + a}{2y_1}, & \text{if } x_1 = x_2, y_1 \neq 0 
\end{cases}$$

A simple computation shows that the coordinates of $R = P+Q = (x_3,y_3)$ are given by the formulas

$$x_3 = k^2 - x_1 - x_2, \quad y_3 = -y_1 + k(x_1 - x_3)$$
A Derive-program for computing the sum of two points \( U \) and \( V \) on an elliptic curve \( y^2 = x^3 + ax + b \mod p \) (the point \( O \) at infinity is denoted by \([p,p]\))

\[
\text{add}(u, v, a, p, k) := \\
\begin{align*}
\text{Prog} & \\
\text{If } u = [p, p] \text{ or } v = [p, p] & \\
& \quad \text{RETURN } u + v - [p, p] \\
\text{If } u \downarrow 1 = v \downarrow 1 \land \text{MOD}(u \downarrow 2 + v \downarrow 2, p) = 0 & \\
& \quad \text{RETURN } [p, p] \\
\text{If } u = v & \\
& \quad k := \text{MOD}((3 \cdot u \downarrow 1^2 + a) \cdot \text{INVERSE}_\text{MOD}(2 \cdot u \downarrow 2, p), p) \\
& \quad k := \text{MOD}((v \downarrow 2 - u \downarrow 2) \cdot \text{INVERSE}_\text{MOD}(v \downarrow 1 - u \downarrow 1, p), p) \\
& \quad a := \text{MOD}(k \downarrow 2 - u \downarrow 1 - v \downarrow 1, p) \\
& \quad [a, \text{MOD}(-u \downarrow 2 + k \cdot (u \downarrow 1 - a), p)]
\end{align*}
\]
A Derive-program for computing the n-th additive power (=multiple) of a point \( U \) on an elliptic curve \( y^2 = x^3 + ax + b \mod p \) (cf. \texttt{basics.dfw} for examples)

```
multiple(u, n, a, p, b_) :=
  Prog
    b_ := [p, p]
    If n < 0
      u := [u\_1, - u\_2]
      n := ABS(n)
    Loop
      If n = 0
        RETURN b_
      If ODD?(n)
        b_ := add(u, b_, a, p)
      u := add(u, u, a, p)
      n := FLOOR(n, 2)
```
Public-key encryption based on the Discrete Logarithm Problem (DLP)

As is well-known, RSA and a number of other cryptosystems use the fact that the integer factoring problem is hard.

A second big group of cryptosystems (Diffie-Hellman key exchange, ELGamal, DSA etc.) is centered around the so-called discrete logarithm problem (or DLP for short), which will be described in the following.

(Note that in both cases it is widely believed, though not proven, that in general there is no easier way of decryption than solving those underlying hard problems in the case at issue!)
The Generalized Discrete Logarithm Problem (GDLP)

Given a finite cyclic group $G$ of order $n$, a generator $g$ of $G$ and an element $h \in G$, find the unique integer $x$, $0 \leq x < n$, such that $g^x = h$. This integer $x$ is called the discrete logarithm of $h$ to the base $g$ and is denoted by $\log_g \beta$.

The most important examples of $G$ are the multiplicative groups of a finite field $\mathbb{F}_q$, where $q$ is either a big prime or a big power of 2 (the „classical“ DLP), or groups emerging from the theory of elliptic curves (ECDLP).
Diffie-Hellman (DH) key exchange in prime residue class groups

If Alice and Bob want to share a secret key $K$, they first agree on a prime $p$ of appropriate size and a generator $g$ of the cyclic group $\mathbb{Z}_p^*$.

Then they perform the following actions:

- Alice chooses a secret key $a$, $0 < a < p-1$, and sends Bob the number $g^a$.
- Bob chooses a secret key $b$, $0 < b < p-1$, and sends Alice the number $g^b$.
- Both can easily compute the shared $K = g^{ab}$, namely Alice by computing $K = (g^b)^a$ and Bob by computing $K = (g^a)^b$. 
Adaption of this DH-scheme for elliptic curves

If Alice and Bob want to share a secret key $K$, they first agree on an elliptic curve $E$ and a point $P$ on it, which has a "big" order $n$.

Then they perform the following actions:

- Alice chooses a secret key $a$, $0 < a < n$, and sends Bob the point $aP$.
- Bob chooses a secret key $b$, $0 < b < n$, and sends Alice the point $bP$.
- Both can easily compute the shared $K = (ab)P$, namely Alice by using the formula $K = a(bP)$ and Bob by using $K = b(aP)$. 
The Generalized Diffie-Hellman problem (GDHP)

Closely related to GDLP is the so-called Generalized Diffie-Hellman problem (GDHP) that was actually used in our example above: Given a finite cyclic group $G$ of order $n$, a generator $g$ of $G$, and group elements $g^a$ and $g^b$ find $g^{ab}$. The GDHP is at most as hard as the GDLP, but it remains an open question, whether they are computationally equivalent in general. (In some special cases this can be proven though, e.g. if the factorization of $n$ is known and $\varphi(n)$ is $B$-smooth, where $B=O((\ln n)^c)$ for some constant $c$.)
The generalized ElGamal-Cryptosystem

Key Generation for A.

- Select an appropriate cyclic group or order n with generator g. (In practice, n has about 160 bits.)
- Select a random integer \( a \) with \( 0 < a < n \) and compute the element \( h = g^a \).
- A’s public key is \((g,h) \in G \times G\), along with a description of the group \( G \). A’s private key is the number \( a \).
The generalized ElGamal-Cryptosystem (cont.)

B, who encrypts a message m for A, should do the following:

- Obtain A’s authentic public key (g, h) along with all necessary information about the group G.
- Represent the message m as an element of the group G.
- Select a random integer k with 0 < k < n.
- Send c = (g^k, mh^k) to A.

A decrypts c = (c_1, c_2) to m by

- computing \( c_2 c_1^{-a} = m(g^a)^k(g^k)^{-a} = m \)
The group G, which is used here, should satisfy the following two conditions:

- **Efficiency**, as regards the computation of group operations.
- **Security**, as regards the DLP for G.

Some groups, which meet these criteria, are:

- The multiplicative group of $\mathbb{Z}_p^*$.
- The multiplicative group of a field $\mathbb{F}_q$, in particular in the case $q=2^m$.
- The group $E$ of point on an elliptic curve over some finite field $\mathbb{F}_q$.
- The so-called class group of an imaginary quadratic number field.
Some adaptations for ElGamal on elliptic curves

Key Generation for A.

- Select an appropriate elliptic curve $E$ and a point $P$ on it of order $n$, where $n$ has again about 160 bits.
- Select a random integer $a$ with $0 < a < n$ and compute the $Q = aP$.
- $A$‘s public key is $(P, Q) \in E \times E$, along with a description of the elliptic curve $E$. $A$‘s private key is the number $a$. 
Some adaptions for ElGamal on elliptic curves (cont.)

B encrypts a message m for A by doing the following:

- Obtain A's authentic public key (P,Q) along with all necessary information about the elliptic curve E.
- Represent the message m as an element of $\mathbb{Z}_p^*$. 
- Select a random integer k with $0 < k < n$.
- Send $c = (kP, mx)$ to A, where x is the x-coordinate of the point kQ and mx is the product in $\mathbb{Z}_p^*$. (As for kQ a so-called point compression can be used, by giving only the x-coordinate of kP as well as the parity of the y-coordinate.)

A decrypts $c = (c_1, c_2)$ to m by $(x, y) = ac_1$ and $m = x^{-1}c_2$. 
How do we get „big“ primes?

The checking, whether a given k-bit prime (where k=160 or greater for ElGamal) is prime is very fast and can be done in $O(k^3)$ time using probabilistic methods (usually by carrying out a fixed number of Rabin-Miller tests as this is also done by Derive internally).
\[ r := \text{random}(2^{160}) \]

Basically, in order to get a random number with at most 160 bits using \text{random}(2^{512}) Derive performs the iteration

\[ s := 2654435721 \cdot s + 1 \mod 2^{32} \]

5 times starting with a 32-bit random seed \( s \) and concatenates all resulting 5 values of \( s \) (in binary representation). Hence, the naive approach

\[ r := \text{random}(2^{160}) \]

yields only \( 2^{32} \) (\( \sim 4.3 \text{ billion} \)) different numbers and they are all totally predictable!
How to create your own random($2^k$)

Hence, for serious cryptographic applications you should create your own k-bit random number $s$. This could be done for $k=160$ e.g. by setting

$$r := 8 + \text{random}(8)$$

as a 4-bit start value and then simplifying

$$0*(r := 16 \cdot r + \text{MOD}($\text{RANDOM}(0, 16)$, 16)) + \text{FLOOR}($\text{LOG}(r, 2) + 1$)$$

39 times using the icon “=“ immediately to left of the input line. (The current bit-length of the generated random number is shown on the screen.)
Statistical tests for randomness

- **Poker test**

  Let $m$ be any positive integer such that for $k:=\lfloor n/m \rfloor$ the condition $k \geq 5 \cdot 2^m$ holds. Now, divide the sequence $s$ into $k$ non-overlapping parts of length $m$. The poker test checks then, whether they occur with about the same frequency. If $n_i$ denotes the absolute frequency of blocks corresponding to the binary representation of $i \in \{0,1,..,2^k-1\}$ then

  $$X = 2^m (n_0^2 + \ldots + n_{2^k-1}) / k - k$$

  approximately follows a $\chi^2$ distribution with $2^m-1$ degrees of freedom.
Tests for randomness (cont.)

- **Runs test**
  Given a binary sequence, let's call a maximal subsequence of $i$ consecutive 0's a gap of length $i$. In a similar way (by exchanging 0 and 1) one can define a block of length $i$. Both are called runs. This test compares the actual numbers of gaps $g_i$ and blocks $b_i$ with the expected ones. The statistics used is
  $$X = \sum \frac{(g_i-e_i)^2+(b_i-e_i)^2}{e_i} \quad (i=1,2,...,k)$$
  where $e_i$ is the expected number of gaps (or blocks) of length $i$, and $k$ is maximal such that $e_k \geq 5$. It approximately follows a $\chi^2$ distribution with $2k-2$ degrees of freedom.
Attacks of GDLP: Shanks‘ Baby-step giant-step algorithm

Using the notations above, this is an algorithm that takes $O(\sqrt{n})$ group operations to compute the solution $x$ of $\alpha^x = \beta$.

1. Set $m \leftarrow \lceil\sqrt{n}\rceil$, $\gamma \leftarrow \beta$ and $i \leftarrow 0$.
2. Compute the values $\alpha^j$, $j=0,1,\ldots,m-1$ and store them in a list.
3. Compare $\gamma$ with elements in the list above. If $\gamma = \alpha^j$ for some $j$, then return $x = im + j$.
4. Set $\gamma \leftarrow \alpha^{-m} \gamma$ and go to step 3.
Attacks of GDLP (cont.): Pollard‘s rho algorithm

This is a Monte Carlo method for solving the GDLP, which makes clever use of the so-called „birthday paradox“. It has the same running time as the baby-step giant-step algorithm, but requires very little storage in contrast. It is currently the best known algorithm to solve the Discrete Logarithm Problems in general groups.
Attacks of GDLP (cont.): Pohlig-Hellman algorithm

Uses the Chinese Remainder Theorem and other reductions to reduce the GDLP to the cases, where the order of a is any prime factor of n, which can be solved then by BSGS or Pollard’s rho method. In particular, it is highly efficient, if n is “smooth“ w.r.t. a relatively small bound B (i.e. if q ≤ B for all prime factors q of n).
Attacks of DLP (cont.): Index Calculus Method

This is by far the best method for the „classical“ DLP (as well as some unsecure variants of ECDLP), where G is the multiplicative group of some field.

There are several versions of it, but all make use of so-called factor bases, similar as the quadratic sieve or the number field sieve for the integer factoring problem. As these methods it is also „subexponential“, i.e. faster than the exponential methods discussed so far.