Sketch of Lecture 36

Wed, 4/29/2020

Elliptic curve cryptography

The idea of Diffie–Hellman (used, for instance, in DH key exchange, ElGamal or DSA) can be carried over to algebraic structures different from multiplication modulo p.

Recall that the key idea is, starting from individual secrets x, y, to share g^x, g^y modulo p in order to arrive at the joint secret $g^{xy} \pmod{p}$. That's using multiplication modulo p.

One other such algebraic structure, for which the analog of the discrete logarithm problem is believed to be difficult, is elliptic curves.

https://en.wikipedia.org/wiki/Elliptic_curve_cryptography

Comment. The main reason (apart from, say, diversification) is that this leads to a significant saving in key size and speed. Whereas, in practice, about 2048bit primes are needed for Diffie–Hellman, comparable security using elliptic curves is believed to only require about 256bits.

For a beautiful introduction by Dan Boneh, check out the presentation:
https://www.youtube.com/watch?v=4M8_Oo7lpiA

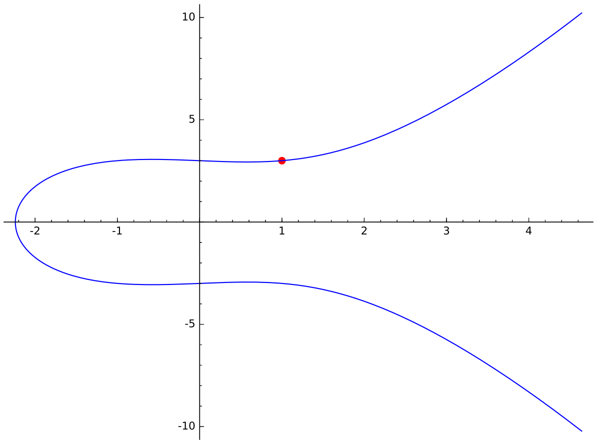
Example 215. The elliptic curve E, described by

$$y^2 = x^3 - x + 9,$$

has the obvious points $(0, \pm 3), (\pm 1, \pm 3)$.

```
Sage] E = EllipticCurve([-1,9])
```

```
Sage] E.plot() + E(1,3).plot(pointsize=50, rgbcolor=(1,0,0))
```



```
Sage] -E(1,3)
```

$(1: -3: 1)$

```
Sage] E(0,3) + E(1,3)
```

$(-1: -3: 1)$

```
Sage] E(0,-3) + E(1,3)
```

$(35: -207: 1)$

```
Sage] E.rank()
```

2

Given a point $P = (x, y)$ on E, we define $-P = (x, -y)$ which is another point on E.
Let us introduce an operation $\boxplus$ in the following geometric fashion: given two points P, Q, the line through these two points intersects the curve in a third point R.

We then define $P \boxplus Q = -R$.

We remark that $P \boxplus (-P)$ is the point O "at ∞". That's the neutral (zero) element for $\boxplus$.
How does one define $P \boxplus P$? (Tangent line!)

Remarkably, the "addition" $P \boxplus Q$ is associative. (This is not obvious from the definition.)
Using $\boxplus$, we learn about many new points: for instance, $(0, -3) \boxplus (1, 3) = (35, -207)$

Easier to verify (but not producing anything new) is $(0, 3) \boxplus (1, 3) = (-1, -3)$.

For cryptographic purposes, elliptic curves are considered modulo a (large) prime p.

Example 216. Let us consider $y^2 = x^3 - x + 9$ (the elliptic curve from the previous example) modulo 7. List all points on that curve.

Solution. Note that, because we are working modulo 7, there are only 7 possible values for each of x and y. Hence, we can just go through all $7^2 = 49$ possible points (x, y) to find all points on the curve.
Doing so, we find 9 points: $O, (0, \pm 3), (\pm 1, \pm 3), (2, \pm 1)$.
[Recall that O is the special point (∞, ∞) which serves as the neutral element with respect to $\boxplus$.]

Comment. A theorem of Hasse–Weil says that the number of points on an elliptic curve modulo p is always close to p. Moreover, we can compute the exact number of points very efficiently.

By taking everything modulo 7, we still have the previously introduced addition rule $\boxplus$.

For instance. $(0, -3) \boxplus (1, 3) = (35, -207) \equiv (0, 3)$

```
Sage] E7 = EllipticCurve(GF(7), [-1,9])
```

```
Sage] E7.points()

   [(0 : 1 : 0), (0 : 3 : 1), (0 : 4 : 1), (1 : 3 : 1), (1 : 4 : 1), (2 : 1 : 1), (2 : 6 : 1), (6 : 3 : 1), (6 : 4 : 1)]
```

```
Sage] E7(0,-3) + E7(1,3)

   (0 : 3 : 1)
```

```
Sage] E7(1,-3) + E7(0,-3)

   (6 : 3 : 1)
```

Multiples of a point are simply denoted with nP. For instance, $3P = P \boxplus P \boxplus P$.

We then have a version of the **discrete logarithm** problem for elliptic curves:

(discrete logarithm) Given P, xP on an elliptic curve, determine x.
(computational Diffie–Hellman) Given P, xP, yP on an elliptic curve, determine $(xy)P$.

Comment. Interestingly, it appears that the computational Diffie–Hellman problem (CDH) is more difficult for elliptic curves modulo p than for regular multiplication modulo p. Indeed, suppose that p is an n-digit prime. Then the best known algorithms for regular CDH modulo p has runtime $2^{O(\sqrt[3]{n})}$, whereas the best algorithm for the elliptic curve CDH modulo p has runtime $\sqrt{p} \approx 2^{n/2} = 2^{O(n)}$.

As a consequence, it is believed that a smaller prime p can be used to achieve the same level of security when using elliptic curve Diffie–Hellman (ECDH). In practice 256bit primes are used, which is believed to provide security comparable to 2048bit regular Diffie–Hellman (DH); this makes ECDH about ten times faster in practice than DH.

Comment. On the other hand, due to that reduced bit size, quantum computing attacks on elliptic curve cryptography, if they become available, would be more feasible compared to attacks on ElGamal/RSA.

Comment. Apparently, more than 90% of webservers use one specific, NIST specified, elliptic curve: P-256:

$$y^2 = x^2 - 3x + 41058363725152142129326129780047268409114441015993725554835256314039467401291,$$

taken modulo $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ (the fact that $p \approx 2^{256}$ makes the computations on the elliptic curve much faster in practice). The initial point $P = (x, y)$ on the curve has huge coordinates as well.

Using this single curve is sometimes considered to be problematic, especially following the concerns that the NSA may have implemented a backdoor into Dual_EC_DRBG, which was a NIST standard 2006–2014.

https://en.wikipedia.org/wiki/Dual_EC_DRBG

A popular alternative is the curve Curve25519. Besides some desirable theoretical advantages, its parameters are small and therefore not of similarly mysterious origin as the ones for P-256:

$$y^2 = x^3 + 486662x^2 + x, \quad p = 2^{255} - 19, \quad x = 9.$$

[Instead of points with (x, y) coordinates, one can actually work with just the x-coordinates for an additional speed-up.]

https://en.wikipedia.org/wiki/Curve25519

```
Sage] E = EllipticCurve(GF(2^255-19), [0,486662,0,1,0])
```

```
Sage] E
```

$$y^2 = x^3 + 486662\,x^2 + x$$

```
Sage] E.order()
```

$$57896044618658097711785492504343953926856930875039260848015607506283634007912$$

```
Sage] log(E.order(),2).n()
```

$$255.000000000000$$

```
Sage] P = E.lift_x(9)
```

```
Sage] P
```

$$(9 : 14781619447589544791020593568409986887264606134616475288964881837755586237401 : 1)$$

```
Sage] 100*P
```

$$(44032819295671302737126221960004779200206561247519912509082330344845040669336 : 4927003\backslash$$
$$82262105253401512144443272943508842110611539588458372871019948920766605 : 1)$$

```
Sage] P.order()
```

$$7237005577332262213973186563042994240857116359379907606001950938285454250989$$

```
Sage] log(P.order(),2).n()
```

$$252.000000000000$$

```
Sage] E.order() / P.order()
```

$$8$$

```
Sage] 5*(20*P) == 20*(5*P)
```

```
1
```

On the other hand, it should be pointed out that it is not an easy task to “randomly generate” cryptographically secure elliptic curves plus suitable base point. That’s the reason pre-selected elliptic curves are of importance.

http://blog.bjrn.se/2015/07/lets-construct-elliptic-curve.html