2014-02-07.sagews

February 10, 2014

Contents

1 February 7, 2014: Elliptic Curves

- on feb 7 mainly use 2014-02-03; but also, something about Diffie-Hellman below.

1.1 Recall Diffie-Hellman

1.1.1 The protocol

- Choose a prime p.
- Choose a base g in $\mathbf{F}_p$.
- Person 1 sends $g^n \pmod p$ and person 2 sends $g^m \pmod p$, where n, m are chosen at random.
- The shared secret is $s = g^{nm} \pmod p$, which both parties can compute.

```
p = next_prime(2^127)
g = Mod(2,p); g
```
2

```
g.multiplicative_order()
```
170141183460469231731687303715884105756

```
p-1
```
170141183460469231731687303715884105756

1.1.2 Attack

- To attack Diffie-Hellman, you solve the discrete log probem in the group generated by g.
- The complexity is the same as attacking this problem in the largest prime divisor of the order of g
- A trivial-to-implement algorithm called baby-step giant-step solves discrete log in any group of order r in time (and space) $\sqrt{r}$.

```
show(factor(g.multiplicative_order()))
```

$$2^2 \cdot 3 \cdot 13 \cdot 23 \cdot 79151 \cdot 54721235939 \cdot 10948250129457457283$$

So in our example, the number of operations needed to solve DL is basically $\sqrt{10948250129457457283}$:

```
# with a really fast computer and good implementation, that's about this \
   many seconds:
N(sqrt(10948250129457457283)) / 1e9
```

```
3.30881400647686
```

1.2 So

Moral: When creating a Diffie-Hellman key exchange, make sure that the group generated by g is of order: (big prime) times (little stuff).

For example, choose $g = 2$ so that it has order $p - 1$ and such that $(p - 1)/2$ is prime. Primality testing is fast, so this is do-able.

1.3 Diffie-Hellman on an elliptic curve

Introduced to the world by our very own Neal Koblitz (and also by Victor Miller at the same time)

- Choose a specific elliptic curve E over a finite field $\mathbf{F}_p$ (same thing as $\mathbf{Z}/p\mathbf{Z}$.
- Choose a specific point $G \in E(\mathbf{F}_p)$.
- Person 1 sends nG and person 2 sends mG, for random n and m.
- The shared secret is the point nmG.

1.3.1 Big problem

- How on earth are you going to know that G has order that is not just a product of small primes?
- This is nothing like testing $(p - 1)/2$ for primality.
- Seems really hard.

```
# Rene Schoof didn't think so...
salvus.file('9aa5a04f7649.jpg')
```

1.3.2 Schoofs idea

- No obvious way to compute $\#E(\mathbf{F}_p)$ directly.
- So sneak up on it, by cleverly computing $\#E(\mathbf{F}_p) \pmod{\ell}$ for many primes ℓ.
- Then, use the Chinese Remainder theorem to obtain the integer $\#E(\mathbf{F}_p)$.

Schoof figured out how to compute $\#E(\mathbf{F}_p) \pmod{\ell}$ efficiently by explicitly computing information about the Frobenius map:

$$(x, y) \mapsto (x^p, y^p)$$

on the subgroup of elements of $\#E(\mathbf{F}_{p^r})$ of order dividing ℓ (Here $\mathbf{F}_{p^r}$ is a sufficiently large finite field.) Full details are well beyond the scope of this course.

But you can try it out!

```
p = next_prime(10^40); p
```

```
10000000000000000000000000000000000000121
```

```
E = EllipticCurve(GF(p), [2,3]); E
```

```
Elliptic Curve defined by y^2 = x^3 + 2*x + 3 over Finite Field of size
10000000000000000000000000000000000000121
```

```
%time n = E.cardinality(); n  # this uses Schoof's algorithm
```

```
10000000000000000000119408859820876289124
CPU time: 0.00 s, Wall time: 0.00 s
```

```
factor(n)
```

```
2^2 * 11 * 41 * 281 * 317 * 622297254123322936910944860793O3
```

Homework: Get a sense of the complexity. Does it get twice as hard to compute cardinality as we had a digit, or polynomial harder? (The claim that the algorithm is fast is the claim that it gets only a bit harder as we add digits.) Also figure out the polynomial in Sage.

1.3.3 Lets try it out Diffie-Hellman on this elliptic curve

```
p = next_prime(10^40); p
E = EllipticCurve(GF(p), [2,3]); E
P = E([7640606313052404727871466222990343759309, \
   2848268726722914253557401365971633049742])  # found using #E.\
   random_point()
```

```
10000000000000000000000000000000000000121
Elliptic Curve defined by y^2 = x^3 + 2*x + 3 over Finite Field of size
10000000000000000000000000000000000000121
```

```
P.order().factor()
```

```
11 * 281 * 317 * 62229725412332293691094486079303
```

```
n = randrange(1,10^40)
```

```
n*P
```

```
(9742018745458282596957305481344155761073 : 3305191629911698505564085013673311187248 : 1)
```

```
# somebody else do this:
#    m = randrange(1,10^40)
#    m*P
```

```
mP = E([ ??? ])
secret = n*mP
```

Coming up how elliptic curves are used in Bitcoin, Playstation, Microsoft DRM, etc