2014-01-27.sagews

January 27, 2014

Contents

1 January 27, 2014: Public-key Cryptography, part 1.

- screencast:
- whiteboard:

1.1 Raising numbers to large powers modulo n

is fast!

```
md("## Diffie-Hellman key exchange demo", hide=False)

# Step 1. Agree on a shared common prime
set_random_seed(0)
p = next_prime(ZZ.random_element(2^512))
print "p=",p
```

1.2 Diffie-Hellman key exchange demo

```
p= 5372617329289865003922404599521029743879407240331442034473769227090926652513391938718622969901716609964397540979209966100873862373968442379743697516525277
```

```
# Step 2. Generate a secret random number
set_random_seed()    # this is harder than it looks :-)
m = ZZ.random_element(p)
print "m=", m
```

```
m= 3975408305143091213989404077423635422991832092015486019732241112895829201039442819215483000499409368854904393133275645921330154632804555617654032049553050
```

```
# Step 3: Tell everybody 2^m (mod p)
t = Mod(2,p)^m
print "2^m (mod p) =", t
```

```
2^m (mod p) = 330965288192958778410906866384385032026721956852465796681697594117503880300
883075866952199486305557357721728936853105386406714332615418085805895850533794
```

```
# Step 4: Compute the shared secret

# Paste other person's t here:
t0 =

secret = t0^m
print "shared secret=", secret
```

1.3 The Discrete Log Problem

Suppose you want to crack a Diffie-Hellman secret, and you are listening on everything that goes over the wire. What can you do?

(Note: In practice one either choses p so that the order of 2 is $p-1$ or replaces 2 by another number of order $p-1$. So we will assume below that 2 has order $p-1$.)

You know:

- the prime p; the number $2^m \pmod p$; the other number $2^n \pmod p$

You do not know:

- the number m and the number n

You want to know:

- the number $2^{mn} \pmod p$

What you can do:

- Given $2^m \pmod p$ and p you can simply try $m = 1, 2, 3, 4, \ldots$ until you find an m that will work. Then you know m and can compute $2^{mn} \pmod p$.
- But now think about how long this could take, since p is big. ("You just wont believe how vastly, hugely, mind-bogglingly big it is. The Hitchhikers Guide to the Galaxy)
- There are much better algorithms that, given $2^m \pmod p$ and p, compute $m = \log_2(2^m \pmod p)$. The world-record is a prime around 2^{530}, which took years of CPU time. See `http://en.wikipedia.org/wiki/Discrete_logarithm_records`

So, when people use Diffie-Hellman (which they do, a lot! see `http://www.ietf.org/rfc/rfc4419.txt`) they use a bigger p. The recommended values for min and max [bits] are 1024 and 8192, respectively.

1.4 The Man in the Middle Attack

If you can listen in on our communications, and you can also alter them, then you can do something dangerous: the man in the middle attack!

The basic idea: instead of the two people agreeing on a single shared secret with each other, they instead each agree on a different shared secret with you.

- You then know both shared secrets.
- You decrypt messages from one person, then re-encrypt them to the other, meanwhile listening in on everything.

Exactly this is deployed in practice on the internet by a major government agency: `http://www.techdirt.com/articles/20131004/10522324753/`

Maybe look at /.ssh/known_hosts in a cloud project.

1.5 Next time

Well talk about RSA, which interestingly solves certain problems directly which Diffie-Hellman doesnt directly, e.g.,:

- digitally signing documents (a problem that comes up with email and bitcoin)
- sending encrypted messages (e.g., email)

Also, RSA mainly relies on the difficulty of integer factorization rather than hardness of the discrete log problem.

The best current public-key cryptosystems use elliptic curves, which we havent got to yet. Well spend some time learning the background for them, then get back to public-key crypto again.