MATH 314 Spring 2018 - Class Notes

4/24/2018

Scribe: Connor Clark

Summary: In today's class we covered the following subjects $\rightsquigarrow$ Dixon's Factoring Algorithm, the Discrete logarithm problem, Diffie Hellman Key Exchange Algorithm and it's steps

0.1 Notes

0.1.1 Dixon's Factoring Algorithm:

Trial Division has a running time of O ($\sqrt{n}$)time , with the optimal value of B$\approx$ $(e^{\sqrt{ln(n)*ln*ln(n)}})$

The running time of Dixon's Factoring Algorithm is O $(e^(\sqrt{ln(n)*ln()*ln(n)}))$, which is sub-exponential time. Is this run time better than $\sqrt{n}$**?**
To find out, take the log of both. $ln(\sqrt{n}) = (\frac{1}{2})*ln(n)$

0.1.2 Trapdoor Functions in RSA

The method of a trap-door function for RSA Decryption is based on multiplication or factoring. One Trapdoor method is the Discrete Log Problem. The Discrete Log Problem is simple on the surface, however implementation this problem is difficult.

$$\alpha = \beta^x (modP)$$

If you know α and X then computing α is easy using modular exponentiation (seems to be easy). Now suppose you know α, $beta$, we would like to find X. If this were a problem being solved in calculus it would be fairly simple as follows:

$$a = b^x \rightarrow ln(a) = ln(b^x) = xln(b) \rightarrow x = \frac{ln(a)}{ln(b)}$$

However, this will NOT work modulo P.
This is the discrete log problem, and it seems to be difficult.

0.1.3 Diffie Hellman: Key Exchange Algorithm

The Key Exchange Algorithm does not allow Alice and Bob to send information, but rather to agree on a specific secret key that can be used in symetric key cryptography.

Steps to Diffe Hellman Key Exchange Algorithm:

1. Alice Picks a large prime number P (200 digits)
2. She finds a primitive root r (mod P)
   *the powers of r produce all of the non-zero residues of mod P.
   *everyone can know P and r (this is similar to public key cryptography)
3. Alice picks a random number (a)
   with $2 \leq$ a $<$ P-1
   can only use this number one time
4. Bob picks a random number (b)
   with $2 \leq$ b $<$ P-1
5. Alice Computes
   A = r^a(mod P)
   and sends it to Bob
6. Bob Computes
   B = r^b (mod P)
   and sends it to Alice
7. Alice Computes
   K $\equiv B^a$ (mod P)
   K $\equiv (r^b)^a$ (mod P)
   K $\equiv (r^b a)$ (mod P)
8. Bob Computes
   K $\equiv A^b$ (mod P)
   K $\equiv (r^a)^b$ (mod P)
   K $\equiv (r^a b)$ (mod P)

$\rightarrow$ Alice and Bob have K.
They can use the first K value as a key for AES.

But why is this secure?
Eve Knows P, r, A, B
She does not know a or b.
$\downarrow$
Computing K seems to be as hard as finding a or b.
$\downarrow$
This is the Diffie Hellman Problem.

0.1.4 Breaking Diffie Hellman

To break this Eve needs to solve $A = r^a$ (mod P) for a discrete log problem.
One method- try all possible values for (a) until one works, this will take O (P) time.

Baby Step Giant Step: a method for computing discrete logs

Suppose you want to solve: a $\equiv r^x$ (mod P)

1. compute N
   N = $\lceil \sqrt{P} \rceil$

2. This results in two tables, one for baby step, and one for giant step

   Baby Steps:

   r^i (mod P)
   for every i,
   1 $\leq$ i $\leq$ N

   **

   Giant Steps:

   ar^{jN} (mod P)
   for every j
   1 $\leq$ j $\leq$ N (mod P)

   **

   **There exists a coincidence between these two tables,
   which tells us

3. $r^i \equiv ar^{jN} \rightarrow$ multiply both sides by $r^{jN} \rightarrow r^{i+jN} \equiv$ a (mod P)

So x = i + jN

How do we know that a coincidence exists between these two tables?

$x < p - 1 \leq N^2$

So we can write x in base N as:

x $\equiv a_0 + a_1 * N$

We are going to find a coincidence with

i = a_0

$i = a_1$

$\rightarrow$ This takes O ($\sqrt{P}$) time instead.

There does exist a faster method that uses tricks like Dixon's Factoring,
it is called Index Calculus.
Index Calculus runs in O($e^{\sqrt{ln(p)*ln*ln(p)}}$) time, which is considered sub-exponential time.