Class Notes, November 10^{th}

Dalton Watts

November 14, 2016

Factoring; We know n is composite. We want to find some $d * e = n$ where $d, e \neq 1, n$

Trial division: Try to divide n by 2, 3, 4, ..., Sqrt(n) and wait 'til you find a factor.

The worst case is that this would take Sqrt(n) steps, which is way too slow.

Basic principle: If $x^2 \equiv y^2 \pmod{n}$

and $x \not\equiv \pm y \pmod{n}$

then n is a composite and $d = gcd(x - y, n)$ is a factor of n which is not 1, nor n.

Ex: $n = 77$

Compute $9^2 \equiv 81 \pmod{77}$

$\equiv 4 \pmod{77}$

$\equiv 2^2 \pmod{77}$

$9^2 \equiv 2^2 \pmod{77}$

$9^2 \not\equiv \pm 2 \pmod{77}$

So, $\gcd(9 - 2, 77)$ is a factor of 77.

Factoring: We know n is composite

Method 1: Choose random integers x in [2,$n - 1$] compute $x^2 \pmod{n}$

If this number is a square when viewed as an integer get a way to factor n.

How fast is it going to be, though?

Choosing x randomly, x^2 is "essentially" a random residue (mod n)

How many squares are there less than n? Take the floor of the square root of n; that is the answer.

Probability a random residue is a square is $\frac{\lfloor\sqrt{n}\rfloor}{n} \approx \frac{1}{\sqrt{n}}$

We expect to have to do this $\sqrt{n}$ times, which is the same as trial division.

Personally, I expect that making a specialized computer with all the primes in a list would be the most effective way to crack RSA.

To make this faster, we use Dickson's factoring method.

Factor n.

Pick a "Small Prime Bound" $B \approx e^{\sqrt{\log(n)}}$

Create a matrix where the columns of this matrix correspond to each of the primes less than B

Step i:
1: Pick x_i randomly in $[\sqrt{n}, n-1]$
2: Compute $S_i \equiv x_i^2 \pmod{n}$
3: Use trial division to factor S_i into primes $\leq B$
4: If we can put into matrix a row with entries that are prime factors of S_i
Repeat until the matrix has more rows than columns. Using linear algebra, find a linear combination of rows that produces a row with all even entries. Use rows $i_1, i_2, ..., i_k$ to do this.
Let $x = x_{i_1}, x_{i_2}, ..., x_{i_k}$
$y = \sqrt{S_{i_1} S_{i_2}, ... S_{i_k}}$
$x^2 \equiv y^2 \pmod{n}$
Usually $x \not\equiv \pm y \pmod{n}$
Factor n.

Example (See Sage Worksheet)
$n = 629$
$B = 12$
He formed it in a table, but as I don't know how in .tex format, I'll translate its results into something I can write down. The left side is the left column, the right side are the values along the table of primes labeled 2, 3, 5, 7, and 11.
$73 = 03001$
$80 = 10101$
$87 = 01010$
$94 = 11100$
$62 = 101100$
$133 = 00011$

He used Linear Algebra to narrow the columns down to 73, 80, and 94.

$73 = 03001$
$80 = 10101$
$94 = 11100$
$Sum = 24202$

$x = 73 * 80 * 94$
$y = \sqrt{2^2 + 3^4 + 5^2 + 11^2}$ (Notice the pattern is $Prime^{SumofColumnValues}$)
$y = 2 * 3^2 * 5 * 11$
$x^2 \equiv y^2 \pmod{629}$
$629 = 17 * 37$
Trial division requires $O(\sqrt{n})$ steps.
Dickson's method requires $O(e^{\sqrt{log(n)*log(log(n))}})$ steps.
That grows slower than n^a for any fixed $a > 0$.
Larger than $(\log(n))^B$ for any B.

Since Dickson, basic ideas have improved.

Quadratic Sieve gives a better way to pick the x_i's and runs faster.

The current best algorithm is the Number Field Sieve, which uses ideas from abstract algebra MATH 369 to speed things up further.

If one day someone totally breaks RSA and it's no longer safe, then what other options are there for Public Key Cryptography?

Factoring is a one-way-function. It's easy to do one way (multiplication), but hard to undo (factoring).

A one way function is any invertible function which can be computed quickly, but inverting it is infeasible for large values.

There's one problem that's been a suggested replacement in such an occurence called the "Discrete Logarithm Problem".

Fix a prime number P.

Suppose $\beta \equiv \alpha^x \pmod{p}$($\alpha, \beta$, and x are integers)

If you know α and x easy to compute β with Modular Exponentiation

If you know α and β, finding x is hard

This is the discrete log with base α

Over real numbers this is easy if $\beta = \alpha^x$

Then we can solve for $x = \log_\alpha(\beta) = \frac{\ln(\beta)}{\ln(\alpha)}$ which is an irrational number. This makes no sense $\pmod{p}$

Naive way to solve this is to try values of x until we find one where $\beta \equiv \alpha^x$

Diffie-Hellman key-exchange

Alice and Bob want to communicate using AES need a key could use RSA, but here's another method:

1: Bob chooses a large prime p and a primitive root $\pmod{p}$

Recall A primitive root mod p is a residue α where α produces all of the residues mod p for different values of i.

2: Alice chooses a secret x where $1 < x < p - 1$

Bob chooses a secret y where $1 < x < p - 1$

3: Alice sends $A = \alpha^x \pmod{p}$ to Bob

Bob sends $B = \alpha^y \pmod{p}$ to Alice

4: Alice computes $B^x \equiv (\alpha^y)^x = \alpha^{xy} \pmod{p}$

Alice computes $A^y \equiv (\alpha^x)^y = \alpha^{xy} \pmod{p}$

Alice and bob both know α^{xy}

Eve knows p, α, α^x, α^y, but doesn't know x or y.

In order to determine Alice and Bob's secret numbers, Eve has to solve the discrete logarithm problem.

Alice and Bob can now use the first 128 bits of α^{xy} as their key for AES.

Alice can't use this to send a message to Bob, but it will allow them to agree on a key secretly.

Baby Example:

$p = 17$ Check that $\alpha = 3$ is a primitive root.
Alices chooses $x = 11$
Bob chooses $y = 15$
$A = 3^{11} \equiv 7 \pmod{17}$
$B = 3^{15} \equiv 6 \pmod{17}$
Alice then computes $B^1 1 \equiv 6^1 1 \equiv 5 \pmod{17}$
Bob then computes $A^1 5 \equiv 7^1 5 \equiv 5 \pmod{17}$