Dixon's Factorization Algorithm

Kobe Luong

April 25, 2020

Attacking RSA

- The attacker needs to factor n=pq, with p and q being prime numbers, in order to find the decryption exponent.
- There is no known "easy" or "fast" way to prime-factorize.
- You can try the "naive" way, which would be to divide n by all numbers up to $n^{1/2}$.
  - The worst-case time-complexity of this would be O($n^{1/2}$).
  - This may not look bad, but n is the amount of bits the computer has to write down.
  - This means n=2^x, so O($n^{1/2}$) = O($2^{2/x}$) = O(e^{c*x}), which is exponential time-complexity and c is $log_2(2)/2$.
- There is another algorithm for prime factorization using a factoring trick.
  - Find two numbers x and y where x $\not\equiv \pm y$ but $x^2 \equiv y^2$ (mod n),
    1. Pick random values of x randomly in $n^{1/2}$ < x < n.
    2. If (x^2 (mod n))$^{1/2}$ is an intger, call it y.
    3. Then $y^2 \equiv$ ((x^c (mod n))$^{1/2}$)^2 $\equiv x^2$ (mod n).

Example: Try to factor n=91

1. Try x=10.
2. $10^2 \equiv 100 \equiv 9$ (mod 91).
3. So $10^2 \equiv 3^2$ (mod 91).
4. So gcd(91, 10-3) = 7 is a factor of 91.

- The chance of randomly choosing a perfect square between 1 and n is about $1/n^{1/2}$.
- If the probability of success is p, then the expected number of trials until success is 1/p. This means that we expect $n^{1/2}$ trials before success because $1/(1/n^{1/2}) = n^{1/2}$.

- This is simply the *expected* number of tries before success, meaning that the worst-case is at least as high as this average-case. The average-case time-complexity of this algorithm is $\Theta(e^{c*x})$. This is exactly the same as trial division.

Dixon's Factorization Algorithm

- Use the same strategy, but don't give up on x just because it didn't produce a perfect square.
  1. You want to factor n.
  2. Pick a bound, "B." This is a bound on the size of the largest prime factor of a number.
  3. Reject numbers whose largest prime factor is bigger than "B."

Example: B=10

- 30 = 2*3*5
- But 34 = 2*17 is not OK because 17>10
- "Fastest" B is approximately $e^{(ln(n)ln(ln(n)))^{1/2}}$, with "n" being the number of primes less than or equal to B. For example, B = 10 and the primes are 2, 3, 5, and 7. This means n=4.

Dixon's Algorithm

1. Pick values $n^{1/2} < a < n$ if the largest prime factor of $b \equiv a^2$ (mod n) is less than B then we keep a(put it in a list). Repeat until you have n+1 different values of "a" that work.
2. - Write down "a" matrix columns that correspond to prime numbers smaller than B.
   - Rows correspond to each of our "a" values.
   - For each a; we record the prime factorization of $b_i \equiv (a_i^2)$ (mod n) by writing down how many times each prime divides.

   Example

   - n = 91, B = 10(2, 3, 5, 7), a = 11
   - b = $11^2 \equiv 30$ (mod 91)
   - 30 = 2*3*5
   - $\begin{array}{cccc} 2 & 3 & 5 & 7 \\ \end{array}$ $\begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}$-> "n+1" rows, 11 -> 30
     "n" columns
   - Linear Algebra tells us that some combination of rows can be added together to get all even entries.

3. Find some such combination of rows
    - Let x be the product of the "a" values associated to these rows.
    - Let Y be the product of the b-values associated to these rows.
    - Exponents on the primes in Y are going to be the numbers we get when adding the rows together. All of these numbers are even.
    - Y is a perfect square!
    - Let y=$Y^{1/2}$ then x^2=$(a_1 * a_2 ... a_k)^2 \equiv (b_1 * b_2 ... b_k)^2$
    - Found x and y where $x^2 \equiv y^2$ (mod n)
    - Most of the time n$\not\equiv \pm$ y (mod n)
    - Then we use the factoring trick to factor n.

Example

- Use Dixon's Algorithm to factor n=629.
- B=12.
- Primes less than 12 are 2, 3, 5, 7, 11.
- N=5
    1. Pick values of "a" between $629^{1/2} < a < 629$ and compute b $\equiv a^2$ (mod n). Check if the largest prime factor of the largest of b is less than B.
    2. Write out Matrix

$$\begin{array}{rc} & \begin{array}{ccccc} 2 & 3 & 5 & 7 & 11 \end{array} \\ \begin{array}{r} 59 \\ 62 \\ \underline{\mathbf{73}} \\ \underline{\mathbf{80}} \\ 87 \\ \underline{\mathbf{94}} \end{array} & \begin{bmatrix} 4 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ \underline{\mathbf{0}} & \underline{\mathbf{3}} & \underline{\mathbf{0}} & \underline{\mathbf{0}} & \underline{\mathbf{1}} \\ \underline{\mathbf{1}} & \underline{\mathbf{0}} & \underline{\mathbf{1}} & \underline{\mathbf{0}} & \underline{\mathbf{1}} \\ 0 & 1 & 0 & 1 & 0 \\ \underline{\mathbf{1}} & \underline{\mathbf{1}} & \underline{\mathbf{1}} & \underline{\mathbf{0}} & \underline{\mathbf{0}} \end{bmatrix} \end{array}$$

    3. Find rows in Matrix we can add to get all even entries
        - x = 73 * 80 * 94
        - Y = $2^2 * 3^4 * 5^2 * 11^2$
        - y = $y^{1/2} = 2^1 * 3^2 * 5^1 * 11^1$
        - x (mod 629), -y (mod 629)
        - (472, 268)
        - x^2 (mod 629), y^2 (mod 629)
        - gcd(629, x-y) = 37
        - 629/37 = 17

- The average-case time complexity of this algorithm is $\Theta(e^{(ln(n)ln(ln(n)))^{1/2}})$