MATH 314 Spring 2018 - Class Notes

04/11/2018

Scribe: Andrew Knickman

Summary:

Discussion of Public Key Cryptography and its first implementation, RSA.

Notes:

Public Key Cryptography

- Public Key Cryptography was invented in the 1970's.
- Main Idea: Find a problem that is easy to do one way, but hard to do in reverse (one-way or trapdoor function).
- Alice performs the easy part of this function, and uses it to make the public key.
- With a public key, Alice can reveal the key to everyone, and everyone can use it to send her messages, but to decrypt messages people need to perform the initial problem in reverse (much harder).

RSA

- The first implementation of public key cryptography was RSA.
- Invented by Rivest, Shamir, and Adleman, hence "RSA."
- Main Idea (like public key) is to find a problem that is easy to compute one way, but difficult to compute in reverse (e.g. Multiply 2 primes to get a 200-digit number, and undo it with factorization).
- Easy Part: Multiply 2 primes, p and q, to get a large number (e.g. 200-digit number)
- Hard Part: Undo prime multiplication with integer factorization, such that $n = pq$, to recover p and q.

Steps of RSA

Alice creates a public key:

- Alice picks 2 large prime numbers, p and q, randomly
  - Recent RSA attacks were possible because primes weren't random enough.
- Alice multiplies $pq = n$.
- Alice picks an exponent e, such that $gcd(e, (p-1)(q-1)) = 1$
- Alice publishes (n, e) as her public key.

If Bob wants to send Alice a message:

- Bob looks up Alice's (n, e) that was published in step 4.'
- Bob encodes his message as a number m.
- Bob computes the ciphertext $C \equiv m^e \pmod{n}$
  - This computation should be fast with modular exponentiation.
- Bob sends C to Alice.

Alice takes C and attempts to solve for m:

- Alice needs to raise C to an exponent that will effectively undo raising m to e.
- Recall Euler's Theorem (Basic Principle of Exponentiation):
  - $x^k \pmod{n} \equiv x^l \pmod{n}$ if $k \equiv l \pmod{(\varphi n)}$
- Alice also needs a number d such that $ed \equiv 1 \pmod{(\varphi n)}$
- Alice computes $C^d \equiv (m^e)^d \equiv (m^{ed}) \equiv m^{(1+k\varphi(n))} \pmod{n}$
  $\equiv m^{1+k\varphi(n)} \pmod{n}$
  $\equiv m(m^{k\varphi(n)}) \pmod{n}$
  $\equiv m(m^{\varphi(n)})^k \pmod{n}$ (which is 1 by Euler's Theorem)
  $\equiv m \pmod{n}$
  - NOTE: $ed \equiv 1 \pmod{\varphi(n)}$ can be written as $ed \equiv 1 + k(\varphi(n))$
- In summary, Alice decrypts C by computing $C^d \equiv m \pmod{n}$

Alice needs to compute d

- $ed \equiv 1$ implies that e and d are inverses, so...
- To get d, Alice computes $d \equiv e^{-}1 \pmod{n}$
  - This computation should be fast with Euclid's Algorithm
- Computing d requires knowing $\varphi(n)$, so how does Alice get $\varphi(n)$?
  - $\varphi(n) \equiv \varphi(pq) \equiv pq(1-(1/p))(1-(1/q)) \equiv (p-1)(q-1)$

- Computing d requires knowing $\varphi(n)$
- Computed $\varphi(n)$ requires knowing p and q
- This requires factoring n (a difficult process)

RSA Example

Suppose Alice picks the prime numbers p = 11, and q = 5.
(In practice, the numbers used should be too big to factor)

- $n = 11 * 5 = 55$ and Alice picks $e = 7$
- Check that e is coprime to $\varphi(n)$ i.e. check if $gcd(e, \varphi(n)) = 1$
- $\varphi(n) = (11 - 1)(5 - 1) = 40$
- Public key is then (55, 7)
- Alice needs d, so she uses Euclid's Algorithm:
  - Compute $gcd(7, 40)$
    $40 = 5(7) + 5$
    $7 = 5(1) + 2$
    $5 = 2(2) + 1$

    $1 = 5 - 2(2)$
    $2 = 7 - 5(1)$
    $5 = 40 - 5(7)$
  - Use substitution to solve for 1
    $1 = 5 - 2(2) = 5 - 2(7 - 1(5)) = -2(7) + 3(5)$
    $= -2(7) + 3(40 - 5(7)) = 3(40) - 17(7)$
  - So $1 = 3(40) - 17(7)$ and the inverse of (7, 40) is -17
    which indicates $d \equiv -17 \equiv 23 \pmod{40}$
  - Alice uses d to decrypt (i.e. this is Alice's Privte Key)
  - Note that there is no need for p and q after Alice computes her private key.

Now Bob wants to send Alice the message m = 2

- Bob first computes $C = 2^7 \pmod{55}$
- Bob then uses modular exponentiation to solve for C
  - $7 = 4 + 2 + 1$
  - $2^1 \equiv 2(mod 55)$
  - $2^2 \equiv 4(mod 55)$
  - $2^4 \equiv 16(mod 55)$
  - So $2^7 \equiv 2^1 * 2^2 * 2^4 \equiv 2 * 4 * 16 \pmod{55}$
- So Bob gets $C \equiv 2(9) \equiv 18 \pmod{55}$

- Then Bob sends $C \equiv 18$ to Alice
- Now Alice takes C and decrypts it by computing $C^d \pmod{n}$

Decryption by Eve

The only way that Eve could decrypt C and recover m is to know d, the decryption exponent. The only way to compute d is to know $\varphi(n)$ because $d \equiv e^-1 \pmod{\varphi(n)}$.
But is there another way for Eve to decrypt C?

- If Eve could find d without knowing $\varphi(n)$ then she could:
  - Multiply $de = 1 + k\varphi(n)$
  - Subtract 1 from both sides to solve for $k\varphi(n)$
  - Then try various small values of k until she found $\varphi(n)$
- Suppose Eve manages to compute $\varphi(n)$ without factoring n.
  - Then she knows n and $\varphi(n)$ where $n = pq$ and $\varphi(n) = (p-1)(q-1)$
  - With this she can compute
    $n - \varphi(n) + 1 = pq - (p-1)(q-1) + 1 = p + q$
  - So with the values of n and $\varphi(n)$, Eve also knows the values of $p + q$ and pq
  - If you know the product and sum of two factors, then you can compute
    $x^2 - (n - \varphi(n) + 1)x + n = (x - p)(x - q)$
    where $(n - \varphi(n) + 1) = (p + q)$ and $n = pq$
- With $(x - p)(x - q)$, you can solve for p and q using the quadratic formula:

  $pq = \frac{(-b \pm \sqrt{b^2 - 4ac})}{2a} = \frac{(n-\varphi(n)+1) \pm \sqrt{(n-\varphi(n)+1)^2 - 4n}}{2}$
- Finding d is as difficult as computing $\varphi(n)$, and computing $\varphi(n)$ is as difficult as factoring n, and nobody knows how to factor n quickly.
- RSA's security lies in the fact that no one can factor n, $\varphi(n)$, and d quickly.